Events and Actions

Last Modified on 10/27/2026 9:04 am CDT

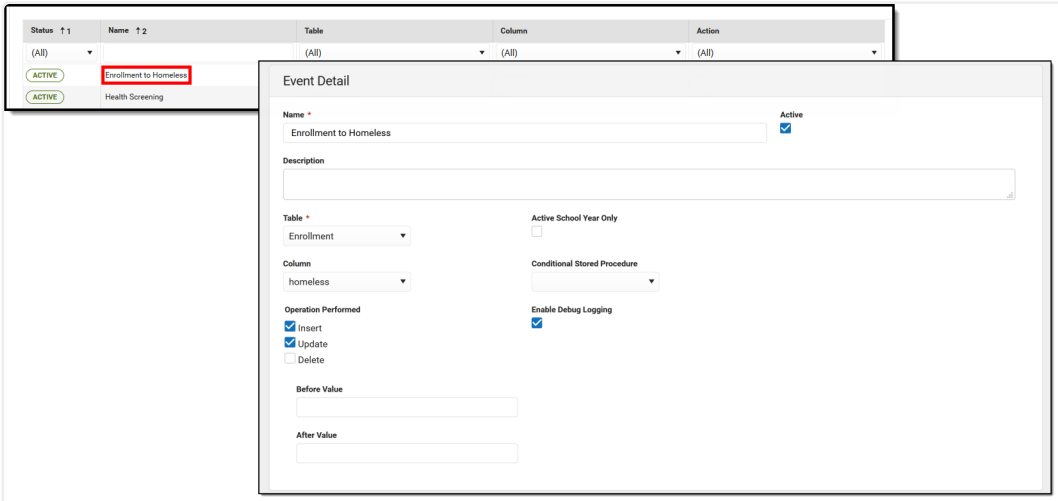
Events & Actions tools are available with Campus Workflow Suite.

Tool Search: Events & Actions Configuration

Events & Actions is a means to cue/prompt staff when a change occurs that needs staff attention, like a schedule change, a new student enrolls in the district, or a student leaves the district. These changes are determined and configured by the district, improving communications among all involved parties.

Users can create an event that will monitor database tables, columns, and column values. Stored procedures can be used to enhance the functionality of this tool, which requires database access.

Watch this Events & Actions video for more information.



Create Events and Actions to track program enrollment, behavior updates, and other student events.

See the Events & Actions Tool Rights article for information on available tool rights.

This tool requires knowledge of SQL, database structures, and the Campus database. If you or your district do not have this sort of knowledge, please contact Infinite Campus to inquire about training, stored procedure creation, or process consulting.

Changes made to database information through Events & Actions may not trigger Campus processes as many are tied to action buttons on the UI (*ex. OLR Processing Workflow*).

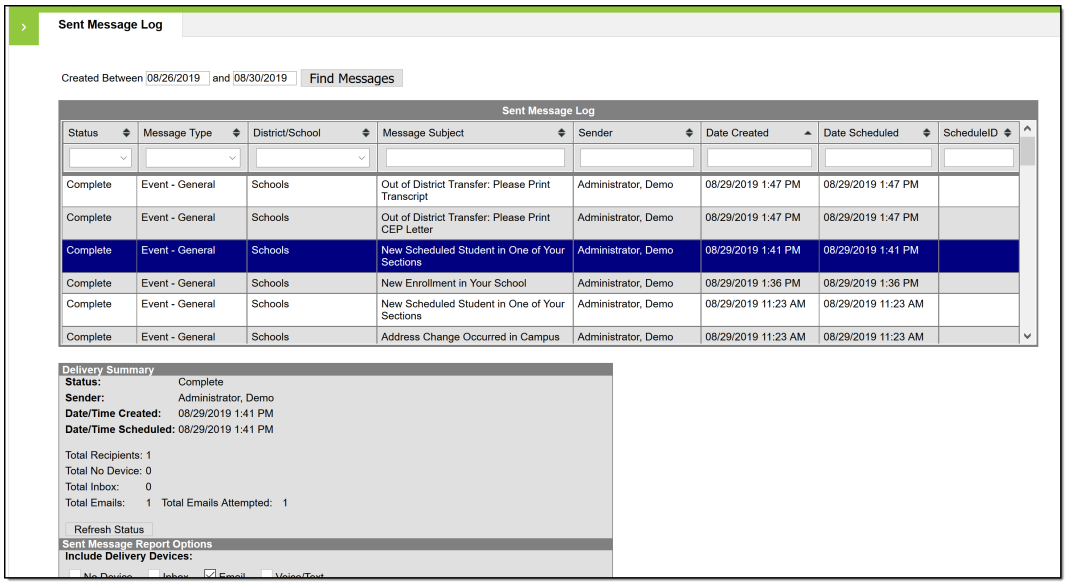
Events and Actions Workflow

Below is the standard workflow that Events and Actions follows.

1. An **Event** is created, along with an associated actions of the event.
2. When the **Event** is made active, a trigger is applied to the database corresponding to the table and column specified in the Event.
3. A change occurs in Infinite Campus that is being monitored by an Event and its trigger is captured.
4. A Quartz job runs on the database every minute and processes the captured events.
5. The Quartz job processes all Actions for the captured Event in the order defined by the Action sequence.

To avoid performance disruptions, Campus recommends saving the first active event for a table during off-peak hours when database activity is low.
Example: No events have been created on the Enrollment table. Consider creating the first event for the table during a time when user activity is low.

As Actions complete, Campus logs a record in the database if the Debug Logging option is marked on the Event editor. If the action is an Email, Campus records an entry in the Sent Message Log and the Recipient Log.



Navigation of Events & Actions

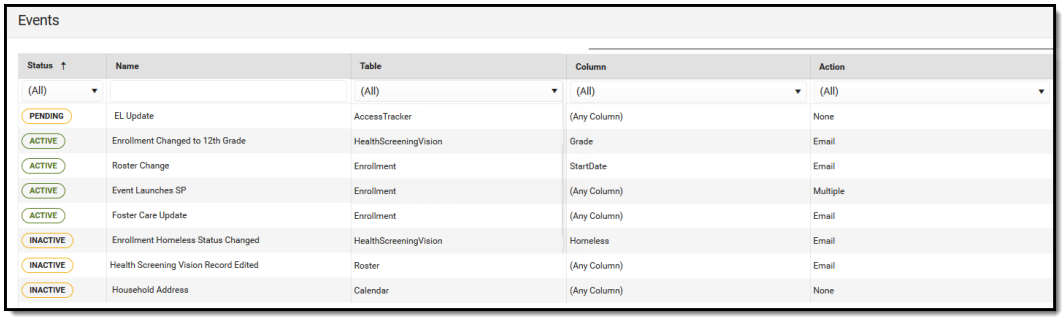
Events & Actions is divided into two editors:

- First, an Event is created. An Event sets the criteria for monitoring an occurrence in the database - a record has been added to a table, a status on an existing record has changed, a record has been deleted.
- Second, an Action (or Actions) is associated with the Event, which is what happens in response to the event. For example, a student ends their enrollment or the building secretary enters an End Date on the enrollment record. That *event* - the ending of an enrollment - requires certain *actions* to occur, like the end dating of the student's locker, alerting/informing the counselor to print a transcript, etc.

Actions can be either Email, which sends information to a user group or a custom list using Campus Messenger functionality, or a Stored Procedure that is created by the district. See the Stored Procedures section for more information on this process.

Events Editor

The Events Editor lists all configured events in various states of completion (active, pending, inactive), and includes the selected database table and column used in the event, and the action taken in response to the Event (email, stored procedure, etc.).



The Events editor is a summary screen to aid users in seeing what has been configured already.

These columns can be resorted as desired, by clicking on the header itself (Status, Name, etc.), or by choosing a value from the dropdown for that column.

Element	Description
Status	Indicates where that event is in the process. Events can be sorted by any of these. Events sort in Pending, Active, Inactive order. • Pending - Event has been created, but the trigger of the event has not been applied or was removed. Essentially, Pending events are waiting for the trigger to be applied to the table.A pending event is similar to an inactive event. While the status is pending, this event and its actions do not happen. • Active - Event is created and the Active checkbox on the Event Detail editor is marked. Any occurrences in the database that meet this Event's criteria will process the associated Action(s). • Inactive - Event is created, but the Active checkbox on the Event Detail editor is not marked. Any triggers that may exist on the associated table will not capture events that meet the criteria of an Inactive Event.
Name	Provides a short summary of the Event, so it can be easily located when modifications need to be made. If the action is an email to building staff that a student's Homeless status changed, the name of the Event might be Homeless Status Changed. On the Detail editor, a Description field is provided so users can add more information to what occurs with the event.
Table	Lists the database table that is defined in the event.
Column	Lists the database column of the selected table that is monitored in the event.
Action	Indicates a summary of the action(s) that occurs when the event is triggered, such as an email is sent to a selected user group.

Select an item to view the Events Detail or click New to create a new event.

Events Detail Editor

The Events Detail editor displays all of the fields necessary to create a new event and establish an action for that event. Provided below are descriptions of the fields on the Events Detail editor.

Field	Description
Name	Provides a short summary of the Event (up to 128 characters. This name displays on the Events Editor (defined above).

Field	Description
Active	Indicates whether the event is active (triggers associated with the event cause the action to occur). If this field is not marked, the event is inactive (triggers associated with the event do not cause the action). The default setting for this field to not marked (inactive). This checkbox needs to be manually marked.
Description	Provides a larger text field (up to 1000 characters) to detail what the event is for and the subsequent actions that are done.
Table	Selection indicates the table the Event is monitoring for change and where the database trigger is placed. Any table in the database can be chosen.
Active School Year Only	Displays only when the selected table includes a CalendarID column. When marked, only events that occur in the active School Year initiate Action(s).
Column	Selection indicates the specific column within a table that defines the exact column from the database that is being monitored by the Event. A column must be selected to enable the Before Value/After Value fields.
Conditional Stored Procedure	Adds an additional condition to the event. When the per minute Quartz Job runs and detects an Event has been triggered, it also checks if a Conditional Stored Procedure has been entered. If the criteria in the Conditional Stored Procedure has been met (is true), then the Actions process for the Event. If the Conditional Stored Procedure is not met (False), then no Actions are processed. If the Conditional Stored Procedure is true, the system checks the condition before each Event Action. An Event Action runs only if the condition remains true at the time that Action is processed. This could be used to associate a second table to an event.
Operation Performed	When the marked option occurs for the selected table, the action (email sent, for example) is processed. If user wants to know when a new enrollment is created for a student, Insert would be selected. If user wants to know when an enrollment field has changed, Update would be selected.
Enable Debug Logging	When marked, event information is recorded to an internal table for troubleshooting purposes. See the Debug Logging table for more information.
Before Value	When a Column is selected, the event conditions are only met when the initial value in the Column is equal to the Before Value. For example, for an event that reports an update to an enrollment field, the before value might be blank; the after value might be Y or 1.
After Value	When a Column is selected, the event conditions are only met when the saved value is equal to the After Value. For example, for an event that reports an update to an enrollment field, the before value might be blank; the after value might be Y or 1.

Event Actions Editor

Following the Events Detail Editor is an Event Actions editor, which lists the actions associated with the selected Event. This displays the Status of the action, the Sequence (because an event can have more than one action) and the name of the action. For example, a change to an enrollment record might have an action of sending an email to building secretaries, and another action to send an email to change the combination on a locker.



When an Action is selected, an Action Detail editor and Action Type editor display.

Action Detail

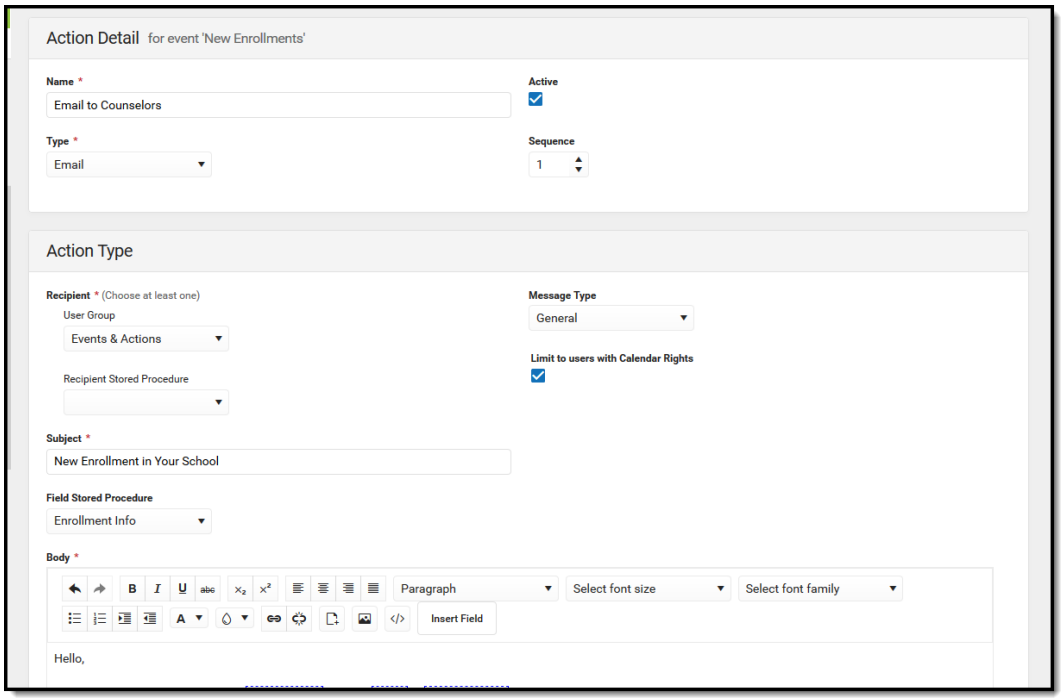
Field	Description
Name	Provides a short summary of the Action (up to 128 characters).
Active	Indicates whether the action is active. When conditions of an event are met, only Actions marked Active will be processed. When this field is not marked, the action is inactive and does not process when the Event conditions are marked. The default setting for this field to not marked (inactive). This checkbox needs to be manually marked.
Type	Indicates whether the action is in the form of an email or a stored procedure. See the following Action Type table for available options. Emails are sent to Campus Messenger for processing. The address used to send emails is entered in the Default Email Sender Address in Email Messenger Settings.
Sequence	Determines the order in which the actions occur. An event could have more than one action. If it does, then sequence numbers the actions in numeric order. If this order needs to be changed after more than one are saved, this value can be modified, or the user can drag and drop the actions into the proper order.

Action Type

This editor varies based on the selected Type in the Detail editor.

Field	Description
Email Type Fields	
Recipient	Determines which staff receive the email. A **User Group** can be chosen, which are created in User Security, or a **Recipient Stored Procedure** can be selected. See the Stored Procedures section for more information. Only one User Group can be selected per action. If multiple user groups should receive an email, use the Copy option to make a copy of the Action and select another User Group. This creates multiple actions for the event that are sequenced and are executed in sequence order. If a user has rights to this tool, they are able to see the list of User Groups, even if they don't have access to user groups in System Administration.

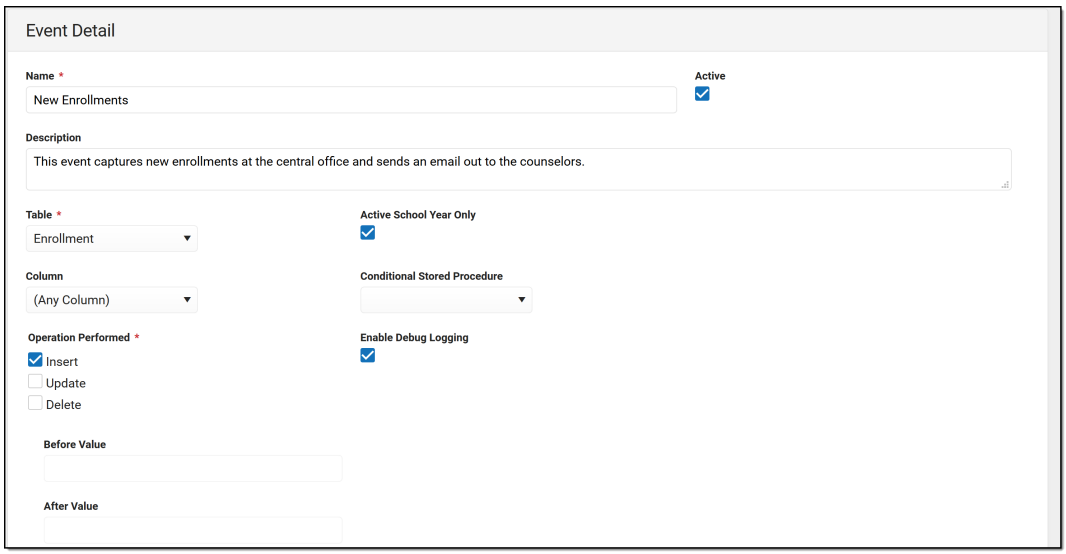
Field	Description
Message Type	Further defines the email recipients. Persons in Campus are assigned Messenger Preferences Contact Reasons. This selection dictates which email address is used for recipients of the email action.
Limit Users with Calendar Rights	This checkbox only appears if the Event table contains a calendarID column. When checked and a record is manipulated in a table with a calendarID, Email actions only send to recipients who have rights to the calendar of the changed record.
Subject	Emails are sent with the entered text.
Field Stored Procedure	Users select the stored procedure that was created to pull specific columns from the Campus database to be displayed in the email body.
Body	Allows creation of an email message using a WYSIWYG editor. Events & Actions uses a new WYSIWYG editor and provides some additional formatting options not available in older WYSIWYG editors. Use the Insert Field option to select the field names to populate specific information for the email. The Insert Field button is only enabled when a Field Stored Procedure has been selected. If the studentName field is selected, the email includes the name of student that requires action.
Stored Procedure Type Fields	
Stored Procedure	Indicates which Stored Procedure that is executed when this event action is processed.



Create a New Event

1. Click the **New** button at the bottom of the Events editor. An **Events Details** Editor opens.
2. Enter the **Name** and **Description** for the Event.
3. Select the **Active** checkbox if the event should be available to run immediately. Otherwise, leave this checkbox not marked.
4. Select the **Table** and **Column** for the Event.
5. Indicate the desired **Operation Performed**.
6. Enter the **Before Value** and **After Value**, if applicable.
7. Mark the **Active School Year Only** checkbox if applicable.

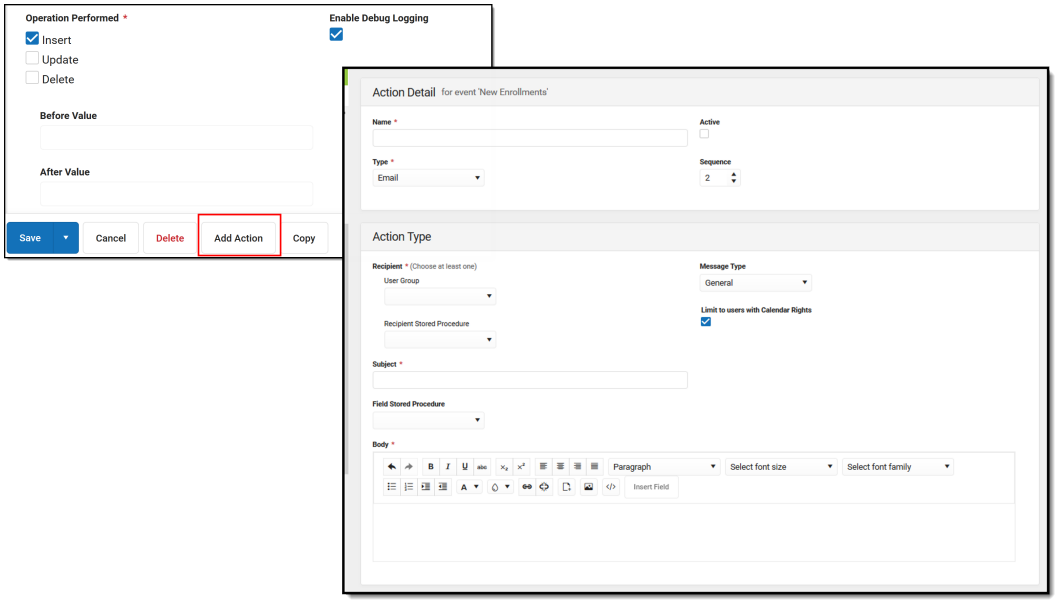
8. Select a **Conditional Stored Procedure,** if desired.
9. Mark the **Enable Debug Logging** to track database messages logged as this event executes.
10. Click the **Save** icon when finished. This saves the event, closes the **Event Detail** editor and displays the **Events** editor.
11. Or, click the arrow next to the **Save** icon to select **Save & Stay**. This saves the event and leaves the Event Detail editor visible so Event Actions can be added.



Create a New Action for the Event

An event must be saved before an action can be created for that event.

1. From the **Events Detail** editor, click the **AddAction** button. An **Action Detail** editor and **Action Type** editor displays.
2. In the Action Detail editor:
   - Enter the **Name** of the Action.
   - Select the **Active** checkbox if the action should be available to run immediately. Otherwise, leave this checkbox not marked.
   - Select the **Type** of action - Email or Stored Procedure.
   - Assign a **Sequence** to the action, or verify the correct Sequence is entered.
3. For an Email Action Type, in the **Action Type** editor:
   - Select the desired **Recipients** to receive a message by choosing a **User Group** or choosing a **Recipient Stored Procedure**.
   - Select the desired **Message Type.**
   - Select the **Limit to Users with Calendar Rights** checkbox if the recipients should be limited to users whose calendar rights match the calendarID of the database record that was changed.
   - Enter the **Subject** of the Email.
   - Select the **Field Stored Procedure** that provides the fields available to the Insert Field button in the WYSIWYG editor.
   - Enter the **Body** of the email in the WYSIWYG editor.
   - Use the **Insert Field**button to customize the email with specific fields provided by the Field Stored Procedure.
4. For a Stored Procedure Action Type, in the New Action Type editor:
   - Select the desired **Stored Procedure** from the dropdown. See the Stored Procedures section for more information.
5. Click the **Save** or **Save & Stay** button. Save returns the user to the Event Detail editor; Save & Stay keeps the user on the Action editors, where an action can be copied or modified.



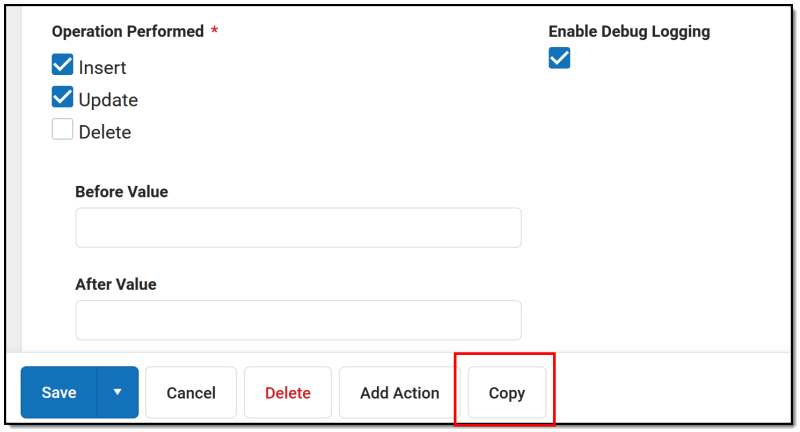
Copy an Event or Action

To copy an event, select an Event from the main Events screen and click the **Copy** button. The Event Detail editor displays. Follow the procedures above for creating an event. The fields are already populated from the event from which the copy was made. Changes can be made to any field.

To copy an action, select the Action from the event and click the **Copy** button. The Action Detail editor displays. Follow the procedures above for creating an action. The fields are already populated from the event from which the copy was made. Changes can be made to any field.

> Upon creation, events and actions are inactive. Users must mark the **Active** checkbox in order to active them.

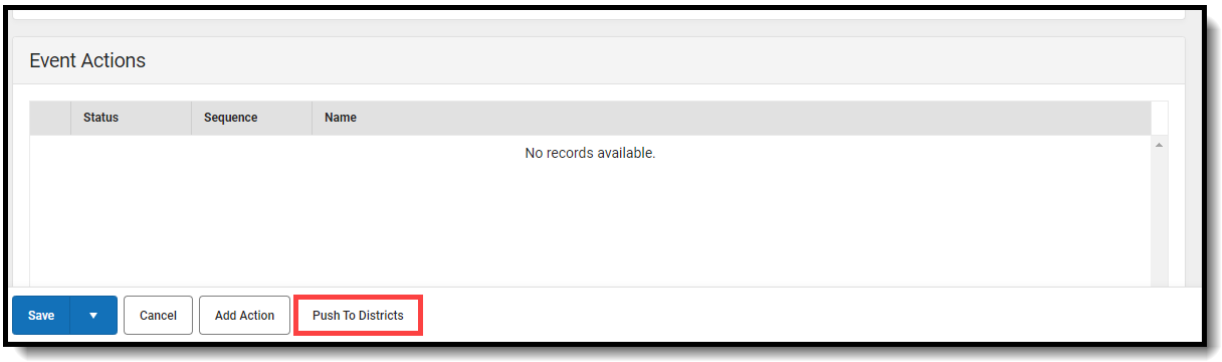
Since only one table and one column can be associated with any event or action, and one user group can be associated for any action, copying events and actions allows different tables to be selected for the same event.



State Edition Events & Actions

States that have purchased the Campus Workflow Suite for the entire state will have access to the Push To Districts tool. When this tool is used, it moves the Events and Actions down to all districts within the state or updates the existing Events and Actions if changes have been made. The Owner column within the Events table lists the State as the owner of that event.

When the state creates a new Event and Action and pushes it down to districts, it is not active by default. Districts must enable the new Events and Actions. After using Push To Districts, there will be a delay before districts see the new Events and Actions.



Events and Actions that have been saved will appear as 'Unpublished' in the Publish Status and the Sync Status before using Push to Districts. The status will change to 'Published' once Push To Districts has been used. When changes are made to an Event or Action, it will again appear as 'Unpublished.' The status will become 'Published' once Push To Districts is used again.



This is only available for state editions that have purchased the Campus Workflow Suite for the entire state.

If a district makes a copy of a state Event and the state makes changes to the original event, the copy will not be updated.

Debug Logging

When the Enable Debug Logging checkbox is marked on the Event editor, records are stored in the database that provide details on why an error occurred.

The following tables provide a list of the possible messages that might occur and some troubleshooting tips to correct it.

These messages are ONLY visible with database access.

Event Errors

Click here to expand...

Event Message	Sample Event Statement	Explanation	Try This...
Unsuccessful Messages			

Event Message	Sample Event Statement	Explanation	Try This...
Event wasn't processed because it was not in the active year	CampusEventLog.addLog(eventLog, "**Event skipped due to non active year**. "); Processing event triggered on Sample table, ID 211728. Event skipped due to non active year.	The event wasn't processed because the action that triggered it did not occur in the active year.	First, verify the event should have occurred in the active year.
Data exception trying to execute SQL from from a stored procedure	CampusEventLog.addLog(log, "**Exception during condition procedure:** " + e.getMessage()); Processing event triggered on Sample table, ID 8294. Exception during condition procedure: Conversion failed when converting the varchar value 'test' to data type int.	The event wasn't processed because there was bad data in the stored procedure.	Review the stored procedure for accuracy.
Stored Procedure executed successfully and failed	CampusEventLog.addLog(log, "**Condition check failed.** "); Processing event triggered on Sample table, ID 8298. **Condition check failed.**	The event wasn't processed because a condition in the stored procedure failed.	Review the stored procedure for accuracy.
Successful Messages			
A Processing Event was triggered	CampusEventLog.addLog(eventLog, "**Processing event triggered on** " + eventRecord.table + " table, ID " + eventRecord.keyID + ". "); Processing event triggered on Sample table, ID 38917. Processing action ID 5 email type. Fetching emails for groupID 152. Found 3 email recipients. Email is given to Messenger for sending.		
Stored Procedure executed successfully and passed	CampusEventLog.addLog(log, "**Condition check passed**. "); Processing event triggered on Sample table, ID 8297. Condition check passed. Processing action ID 127 email type. Fetching emails for groupID 283. Filtering email by calendar rights. Found 0 potential email recipients. Email given to messenger for sending.		

Action Errors

Click here to expand...

Action Message	Sample Action Statement	Explanation	Try This...
Unsuccessful Messages			
Error occurred trying to execute the Action and specifically the wrong action type	CampusEventLog.addLog(log, "CampusEventAction " + actionID + " **references non-existent event handler** " + eventHandler + ". "); Sample message: Processing event triggered on HealthScreeningVision table, ID 8294. CampusEventAction 122 references non-existent event handler email1.	The action wasn't processed because of an incorrect action type.	Review the action associated with the event and modify the action type.

Action Message	Sample Action Statement	Explanation	Try This...
Error executing the stored procedure in the Action condition	CampusEventLog.addLog(log, "**Exception during condition procedure:** " + e.getMessage()); Sample Message: Processing event triggered on Sample table, ID 8294. Processing action ID 121 sql type. Exception during execution of action procedure wf_a_bad: A result set was generated for update.SQL procedure wf_a_bad executed.	The action wasn't processed because a condition in the stored procedure failed.	Review the stored procedure for accuracy.
A result set was generated for...	This is a SQL controlled statement not IC statement Processing event triggered on Sample table, ID 8294. Processing action ID 121 sql type. Exception during execution of action procedure wf_a_bad: A result set was generated for update.SQL procedure wf_a_bad executed.	The action wasn't processed because a condition in the stored procedure failed.	Review the stored procedure for accuracy.
Successful Messages			
Start processing action(s)	CampusEventLog.addLog(log, "**Processing action ID** " + this.actionID + " " + eventHandler + " type. "); Sample message: Processing event triggered on Sample table, ID 38917. Processing action ID 5 email type. Fetching emails for groupID 152. Found 3 email recipients. Email is given to Messenger for sending.		

Email Actions

Click here to expand...

Action Message	Sample Action Statement	Explanation	Try This...
Unsuccessful Messages			
Field Stored Procedure	CampusEventLog.addLog(log, "**Exception during execution of template procedure** " + action.templateProc + ": " + e.getMessage()); Sample Message: Processing event triggered on Sample table, ID 8300. Processing action ID 130 email type. Exception during execution of template procedure wf_t_bad: Conversion failed when converting the varchar value 'test' to data type int. Fetching emails for groupID 283. Filtering email by calendar rights. Found 0 potential email recipients. Email given to messenger for sending.	The process failed because of bad data, i.e., there were no users in the selected user group or a value couldn't be read.	There could be multiple reasons for a conversion failure with data. Review the selected user group for individuals and email addresses.

Action Message	Sample Action Statement	Explanation	Try This...
Recipient Stored Procedure	CampusEventLog.addLog(log, "**Bad recipient procedure name** " + action.recipientProc + ": " + e.getMessage()); Sample Message: Processing event triggered on Sample table, ID 8299. Processing action ID 129 email type. **Bad recipient procedure name %--'_$`&[]: Table name rejected due to possibility of SQL injection: %--'_$`&[]**Error sending email, skipping.	The process failed because of bad characters (brackets, dollar signs, other special characters) in the stored procedure.	Modify the procedure to not include the bad characters.
No email recipients configured	CampusEventLog.addLog(log, "**No email recipients configured!** "); Sample Message: Processing event triggered on Sample table, ID 8296. Processing action ID 125 email type. No email recipients configured! Email given to messenger for sending.	There were no email addresses found; however, the email was still sent to messenger for processing and completion.	Verify staff email addresses are entered for the desired group of users.
Error sending email for (email) skipping	CampusEventLog.addLog(log, "**Error sending email, skipping**. "); Sample Message: Processing event triggered on Sample table, ID 8299. Processing action ID 129 email type. **Bad recipient procedure name %--'_$`&[]: Table name rejected due to possibility of SQL injection: %--'_$`&[]**Error sending email, skipping.	The process failed because of bad characters (brackets, dollar signs, other special characters) in the stored procedure.	Modify the procedure to not include the bad characters.
Exception during execution of action procedure	CampusEventLog.addLog(log, "**Exception during execution of action procedure** " + action.actionProc + ": " + e.getMessage()); Sample Message: Processing event triggered on Sample table, ID 42719. Processing action ID 97 sql type. Exception during execution of action procedure wf_a_createHIbehaviorForm:Conversion failed when converting date and/or time from character string.SQL procedure wf_a_createHIbehaviorForm executed.	This is a SQL error that occurs when there is an issue with the stored procedure.	Review the stored procedure for accuracy.
Successful Messages			
Recipient User Group	CampusEventLog.addLog(log, "**Filtering email by calendar rights**. "); Sample message: Processing event triggered on Sample table, ID 38925. Processing action ID 25 email type. Fetching emails for groupID 285. Filtering email by calendar rights. Found 2 potential email recipients. Email given to messenger for sending.		
Found X potential email recipients	CampusEventLog.addLog(log, "**Found " + recipientList.size() + " potential email recipients.** "); Sample Message: Processing event triggered on Sample table, ID 213760. Processing action ID 91 email type. Fetching emails for groupID 283. Found 2 potential email recipients. Email given to messenger for sending.		

Action Message	Sample Action Statement	Explanation	Try This...
Email given to messenger for sending	CampusEventLog.addLog(log, "**Email given to messenger for sending.** "); Sample message: Processing event triggered on Sample table, ID 38925. Processing action ID 25 email type. Fetching emails for groupID 285. Filtering email by calendar rights. Found 2 potential email recipients. Email given to messenger for sending.		

SQL Actions

Click here to expand...

Action Message	Sample Action Statement	Explanation	Try This...
Unsuccessful Messages			
Exception during execution of action procedure	CampusEventLog.addLog(log, "**Exception during execution of action procedure** " + action.actionProc + ": " + e.getMessage()); Sample Message: Processing event triggered on Sample table, ID 42719. Processing action ID 97 sql type. Exception during execution of action procedure wf_a_createHIbehaviorForm:Conversion failed when converting date and/or time from character string.SQL procedure wf_a_createHIbehaviorForm executed.	This is a SQL error that occurs when there is an issue with the stored procedure.	Review the stored procedure for accuracy.
Successful Messages			
SQL procedure (Name) executed	CampusEventLog.addLog(log, "**SQL procedure " + action.actionProc + " executed**. "); Sample Message: Processing event triggered on Sample table, ID 38754. Processing action ID 98 sql type. SQL procedure wf_a_createHIbehaviorForm executed.		

Event with Multiple Actions

Click here to expand...

An event with multiple actions continues to append the log text as the actions are executed. Successful sample messages for these types of events are below:

Message Sample One:

Processing event triggered on Sample table, ID 8292. Processing action ID 24 email type. Fetching emails for groupID 283. Filtering email by calendar rights. Found 0 email recipients. Email sent. Processing action ID 26 email type. Fetching emails for groupID 283. Filtering email by calendar rights. Found 0 email recipients. Email is given to Messenger for sending.

Message Sample Two:

Processing event triggered on Sample table, ID 8294. Processing action ID 92 email type. Fetching emails for groupID 283. Filtering email by calendar rights. Found 0 email recipients. Email sent. Processing action ID 93 email type. Fetching emails for groupID 283. Filtering email by calendar rights. Found 0 email recipients. Email sent. Processing action ID 94 email type. Fetching emails for groupID 283. Filtering email by calendar rights. Found 0 email recipients. Email is given to Messenger for sending.

For each Action tied to an Event (trigger), the Event (trigger) will be checked. This means if an Event has three actions tied to it, the Event will be checked three times (once for each action). If 10 records are changed that qualify for that Event, it will be checked 30 times.

Events and Actions Scenarios

This section provides a few examples of events and actions that can be created.

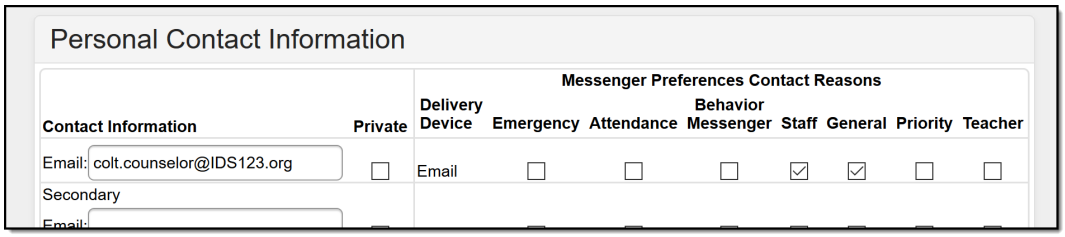
Scenario 1: Enrollment Change

This example is an Event that sends an email to a specific user group when a new enrollment has been added for a student in the current school year.

Step 1. Verify the staff to whom the message is being sent have email addresses entered

- Navigate to **Census > People > Demographics** and search for the individuals who are to receive this email.
- In the **Messenger Preferences Contact Reasons** section, verify the email has been entered and the appropriate Contact Reasons are selected. The Action editor contains a Message Type field; individuals who have a matching Messenger Preferences Contact Reasons checkbox marked receive a message.

For this example, the Counselor has his school email address entered and is marked for Staff and General Message Types.

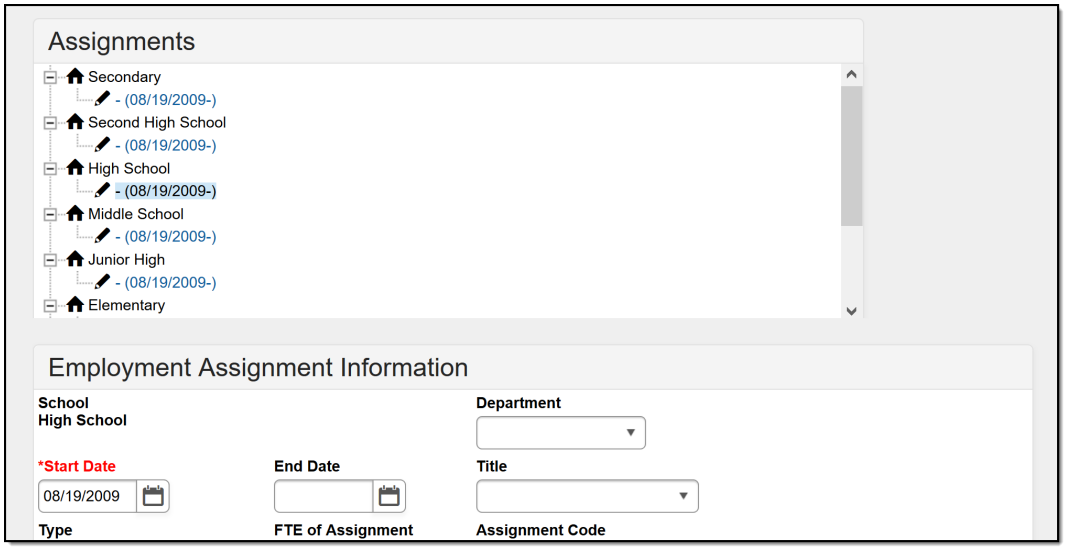


Staff Email and Contact Reasons

Step 2. Verify the staff person has an active District Assignment record

This step won't affect the delivery of the message, but it's a good idea to review assignment records for accuracy.

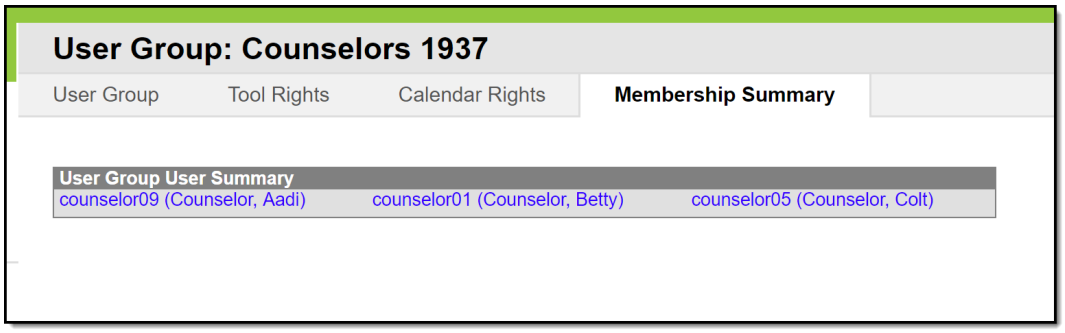
- Navigate to **Census > People > District Assignments** and view the individual's list of assignment and start dates.



Step 3. Verify the correct individuals are included in the User Group that is used in the Event Action

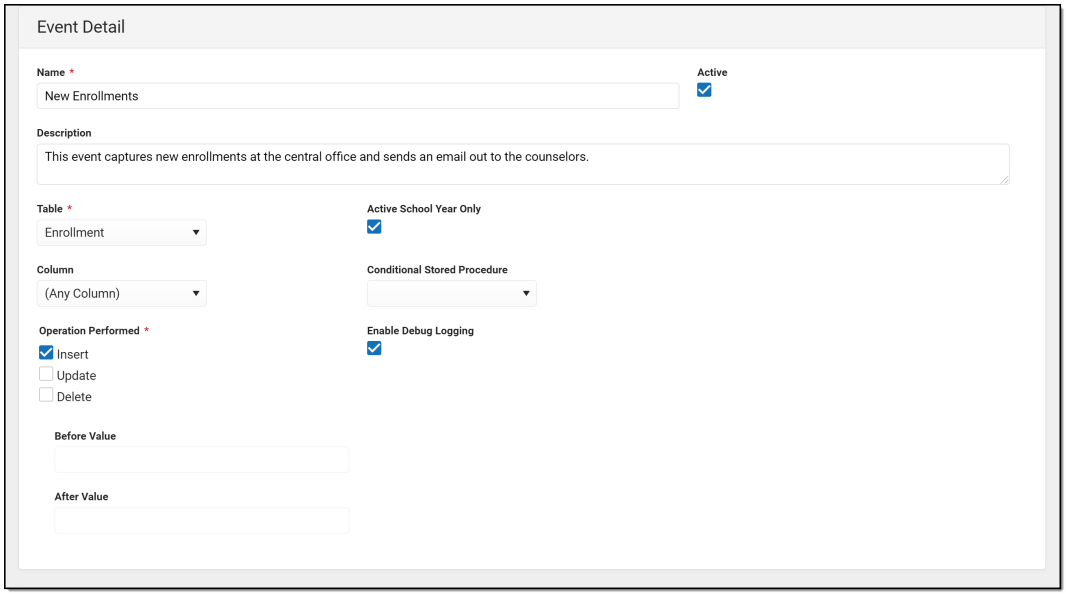
- Navigate to System Administration > User Security > User Group
- Enter the name of the User Group in the Search field and select the desired group.
- Select **Membership Summary**.

Three counselors are included in the group, and all three of them should receive the email about a new enrollment being added.



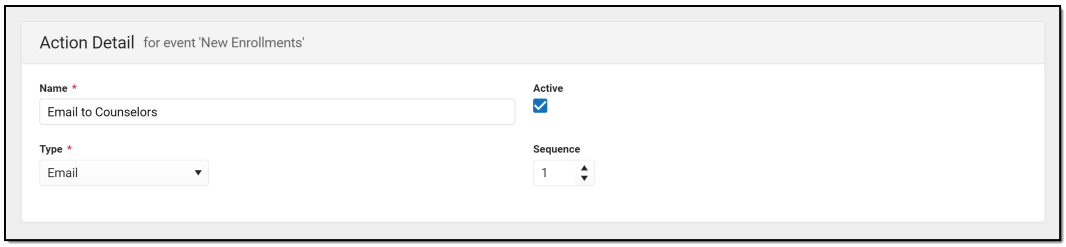
Step 4. Create the Event

- Navigate to **System Administration > Events & Actions > Events & Actions Configuration** .
- Click the **New** button.
- On the **Event Detail** editor, enter the **Name** of the Event. Since this example is to alert counselors of a new enrollment, the Event Name could be **New Enrollments**.
- Enter a more detailed **Description** of the event, perhaps including who receives it.
- Mark the **Active** checkbox.
- Since this is an event related to enrollments, select **Enrollment** from the Table dropdown.
- Mark the **Active School Year Only** checkbox. This only sends an email when the new enrollment is for the active year.
- The **Column** field can be left to the default selection of **(Any Column)**, or insert a specific column. This event isn't looking for a change to an existing column so this isn't necessary.
- For **Operation Performed**, mark **Insert**. This event is looking for a new record in the table.
- Mark the **Enable Debug Logging** for troubleshooting (if access to the database is available).
- The Before Value and After Value fields are not activated since there is no column selection.
- Click the **Save & Stay** button.



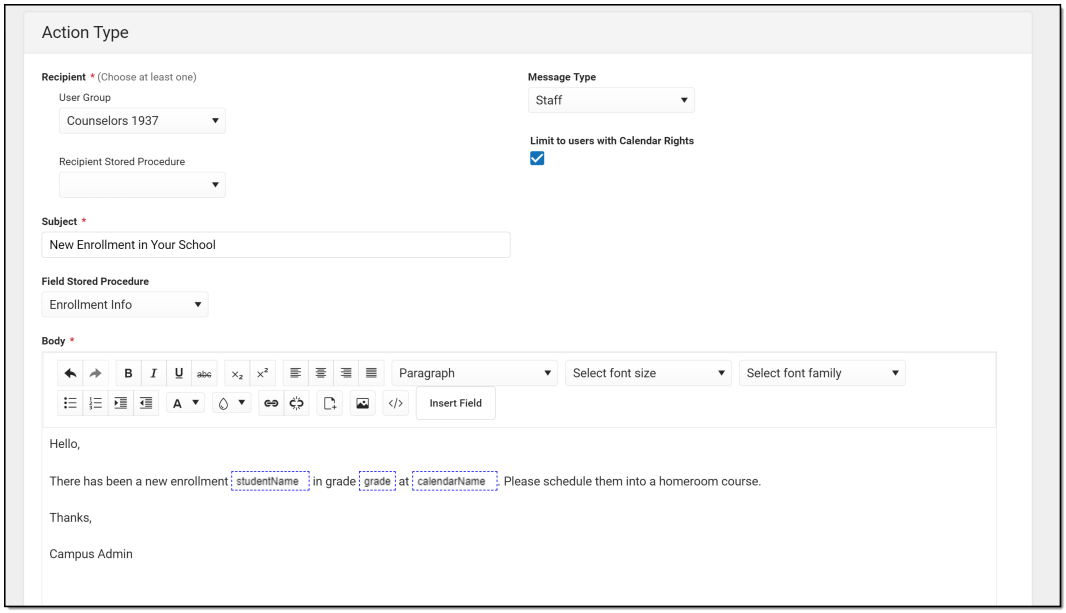
Step 5. Add an Action to the Event

- Click the **Add Action** button.
- On the **Action Detail** editor, enter the **Name** of the Action. Since this example is to alert counselors of a new enrollment, the Name could be **Email to Counselors**.
- Mark the **Active** checkbox.
- Select the **Type** of Email.
- This event has one action, so the **Sequence** can be left at 1.



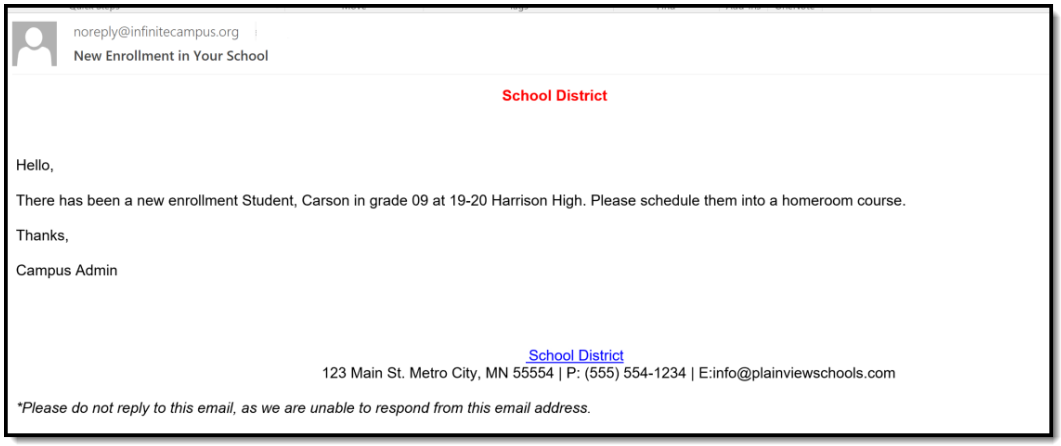
Step 6. Create the Email

- On the Action Type editor, select the **User Group** to receive the email.
- Select the desired Message Type. The counselors in this group have the Staff checkbox marked for their staff emails (marked in Step 1).
- Mark the **Limit to users with Calendar Rights** so only the counselors with calendar rights in the calendar in which the enrollment was created receive an email.
- Enter a **Subject** for the email.
- From the **Field Stored Procedure** dropdown, select the Enrollment Info button. This is a stored procedure that has been added to the database and includes three fields that are used in the body of the email. An email can be sent without these fields, but it won't be customized with the student's name, grade level and calendar name.
- Type the email message in Body field, using the available formatting options of the WYSIWYG editor.
- To customize the message, click the Insert Field button and select the field names.
- Click the **Save** button.



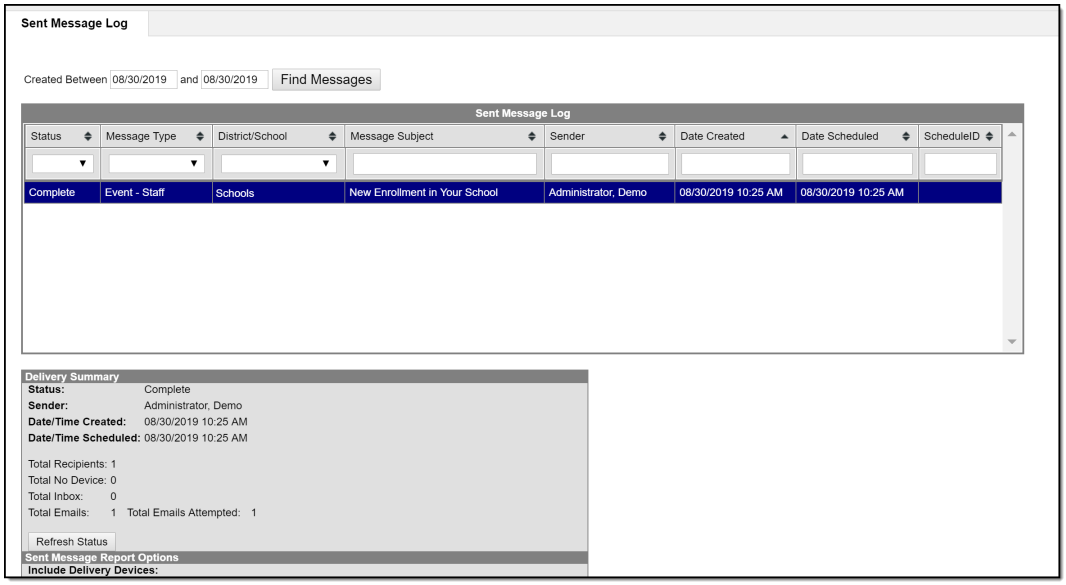
Step 7. New Enrollment Added

The Central Office has added a new enrollment for a student who has transferred to the district from out of state. A database trigger is watching the Enrollment table and captures this insert. The Events & Actions logic records this change in a file and waits for a quartz job to begin processing (runs every minute). The quartz job begins processing the actions for this captured event and the Recipients receive their email.



Step 8. Review the Sent Message Log

In the Sent Message Log, the Central Office or the Administrator can verify that the email was sent and the event was completed.



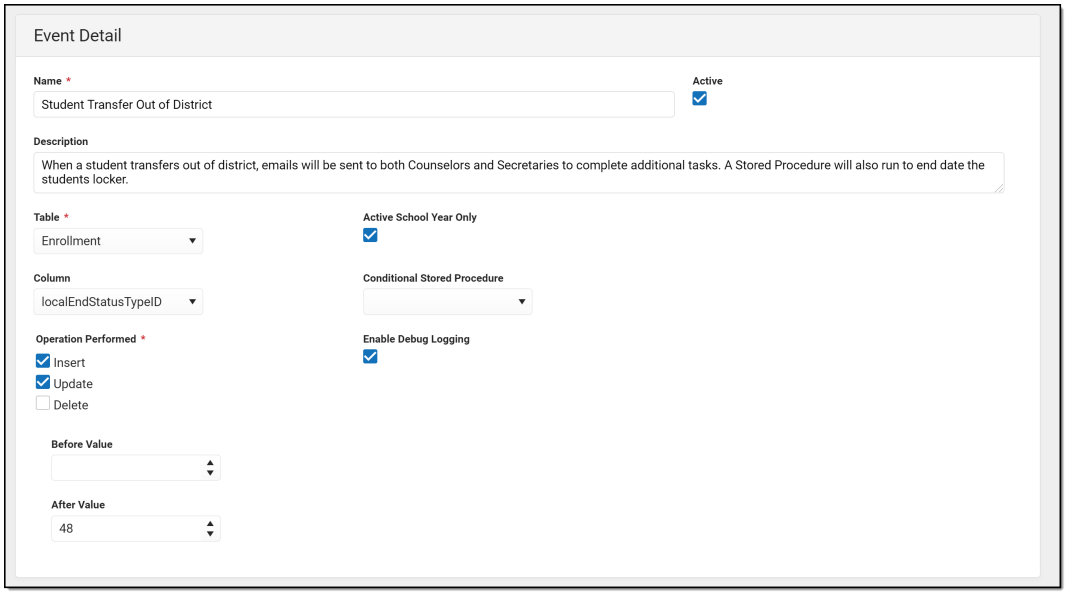
Scenario 2: End Date a Locker

This example is an Event that notifies individuals that a student has transferred out of the district. It uses a stored procedure that has been created by the district to auto-end a student's locker when the enrollment ends.

Assumptions have been made for this example that User Groups are established already and staff emails have been added to Messenger Preferences Contact Reasons and the selections are correct (see Steps 1-3 in the previous example if necessary).

Step 1. Create the Event

- Navigate to **System Administration > Events & Actions > Events & Actions Configuration** .
- Click the **New** button.
- On the **Event Detail** editor, enter the **Name** of the Event. Since this example is to alert counselors of a student transferring out of the district, the Event Name could be **Student Transfer Out of District**.
- Enter a more detailed **Description** of the event to include information about the automatic ending of a student's locker.
- Mark the **Active** checkbox.
- Since this is an event related to enrollments, select **Enrollment** from the Table dropdown.
- Mark the **Active School Year Only** checkbox. This means the enrollment must be ended in the active School Year to initiate an Action.
- For this event, the trigger is related to either an insert or a change for a specific column that is looking for an enrollment end status of a particular type. The **Column** field needs to know which specific table column to look for, so a selection of **localEndStatusTypeID** is selected.
- For **Operation Performed**, mark Insert and Update. This event is looking for a new record in the table or an update to an existing record in the table.
- Mark the **Enable Debug Logging** for easier troubleshooting.
- The **Before Value** field can be left blank, but the **After Value** field needs an entry that reflects the End Status that is being monitored. In this example, the Transfer Out of District End Status has a column value of 48.
- Click the **Save & Stay** button.

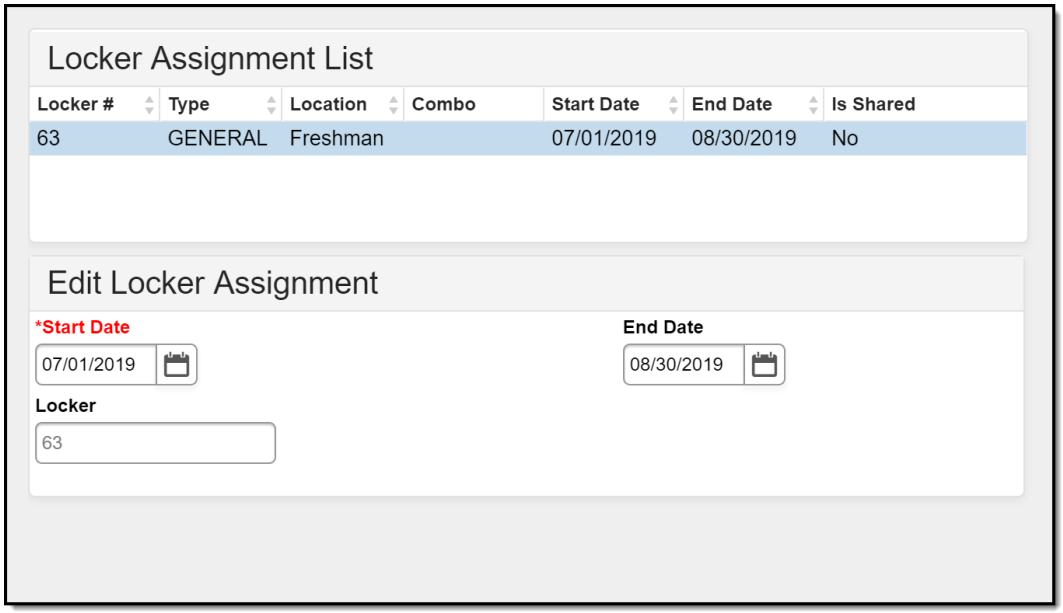


Step 2. Add the Action

This event has two other actions already created - one is an email to Counselors to print the student's transcript, and one is an email to the administrator to print a letter. This new action is to auto-end date the student's locker so it can reassigned to other students as needed.

- Click the **Add Action** button.
- On the **Action Detail** editor, enter the **Name** of the Action. Since this example is to end locker assignments, the Name could be **End Lockers.**
- Mark the **Active** checkbox.
- Select the **Type** of **Stored Procedure**.
- This event has three actions, so the **Sequence** is three. Actions are performed in the sequence order.
- On the Action Type editor, select the appropriate **Stored Procedure.** This has already been tested on the district's staging site, added to the database, and made available in the Action Detail editor. See the Stored Procedures section for more information.
- Click the **Save** button.

When the quartz job runs the actions associated with the Event and are processed in sequence order. The first two actions deliver emails via the Messenger system. The third action runs the stored procedure which end dates the student's locker.



Stored Procedures

Events & Actions allows customers to perform custom tasks when specified changes have occurred in the Campus database. Common options or tasks are available directly in the configuration tool, but to achieve maximum flexibility, some parts of the system can be configured by user-provided stored procedures.

Events & Actions do not require the use of stored procedures; different types of notifications can still be sent as an email. But, the personalized details of that communication (staff name, student name, etc.) would not be included. Without the stored procedure, it would be a generic message.

There are advantages to using Stored Procedures in events:

- Stored procedures can be used in Events to fine tune the exact database change being monitored by the event.
- Stored procedures can be used in Actions with the type of Email to define the recipient list or to provide fields from the database to the email body.
- Stored procedures can be used in Actions with type of Stored Procedure to process a user defined change.

The following information provides guidance on stored procedures. As noted above, if you do not have access to your database to add stored procedures or are unfamiliar with databases, contact Infinite Campus for options.

Click here to expand...

Types of Stored Procedures

There are currently four places where stored procedures can be used. Each use has a different purpose and design.

Type	Description	Example
Conditional	These stored procedures are used to filter events when it is defined by a complex condition not described by a simple change of a database column.	An action is desired when a behavior event is reported for students who participate in a particular program.
Recipient	These stored procedures are used in email actions to select recipients of the email for the particular event.	An email is sent to Primary teacher of a course section when a new student is scheduled into that section.

Type	Description	Example
Field	These stored procedures provide data that is included in the body of an email that is sent for a particular event.	The email needs to contain the school name to inform an administrator that the event pertains only to a particular school.
Action	These stored procedures are used by the stored procedure action type to perform some custom database changes when an event has occurred.	A transportation end date needs to be entered when an enrollment has ended.

Add a Stored Procedure to Events & Actions

Create the stored procedure

A database administrator needs to create the stored procedure in the database and given a name that identifies a procedure and its use.

If the procedure is custom to the district, the procedure should be located in the **cust** schema. This avoids potential future collisions of procedure names and database changes that are made by Infinite Campus.

- Include the site or district abbreviation in the name to assist in identifying the procedure's owner.
- Include briefly the subject of the procedure and its type.

Procedure names should begin with a letter and contain only letters, numbers and underscores as this allows simpler SQL syntax for referencing. An example would be a procedure named 'PSD123_InsertedCriticalBehaviorRole_Field'. This indicates it is a procedure to retrieve email fields for critical behavior roles in the district.

Input parameters

All stored procedures must take the following parameters and types:

@IC_EVENT_ACTION_TYPE	CHAR(1)	This parameter is I, U, or D. • I indicates the event was caused by an insert • U indicates the event was for an update • D indicates the event was for a delete
@IC_EVENT_TIMESTAMP	DATETIME	This parameter is a timestamp that indicates when the database recorded the change.
@IC_EVENT_TABLE	VARCHAR(128)	This parameter contains the name of the table that was changed.
@IC_EVENT_COLUMN	VARCHAR(128)	This parameter contains the name of the column that was changed (if configured).
@IC_EVENT_KEY_ID	INT	This parameter contains the key if the primary key on the changed table is a single INT-datatype column.
@IC_EVENT_PERSON_ID	INT	This parameter contains the personID if the changed table contains a column named personID.
@IC_EVENT_CALENDAR_ID	INT	This parameter contains the calendarID if the changed table contains a column named calendarID.
@IC_EVENT_SCHOOL_ID	INT	This parameter contains the schoolID if the changed table contains a column named schoolID.

@IC_EVENT_DISTRICT_ID	INT	This parameter contains the districtID if the changed table contains a column named districtID.
@IC_EVENT_DATA	VARCHAR(MAX)	This parameter contains a string with a JSON object that contains structured before and after information about the changed record. The object has two fields - one field labeled **old** and one field labeled **new**. • The old field is an object that includes fields that match the changed table's fields. The values for those fields are those that exist in the changed database record before an update or delete. • The new field is an object that includes either fields whose values changed in an update, or those fields that exist in the changed database record in the case of an insert. Example: `{ "old" : { "keyID" :1, "field2" : "hello" , "field3" : false , "field4" : "2019-01-01T23:00:00" , "field5" : null } "new" : { "field2" : "goodbye" , "field3" : true } }`
@IC_EVENT_USER_ID	INT	This parameter contains the userID field from the UserAccount table of the Campus user that changed the record if the change was initiated within most parts of the Campus application.
@IC_EVENT_TOOL_ID	VARCHAR(50)	This parameter contains the toolID or code field from the CampusTool table of the Campus tool that changed the record if the change was initiated within most parts of the Campus application.

Required Output

- **Conditional** procedures must produce a result set containing a single record with a single value of 1 or 0 (as a bit or integer data type). Values of 1 indicate the event continues to process and actions continue to run. All other results cause the event to stop being processed and actions are skipped.
- **Recipient** procedures must produce a single result set that contains a single column of userID from the UserAccount table of potential recipients.
- **Field** procedures must produce a result set with a single record. To allow the user interface to name the fields in the email body WYSIWYG editor, the result set should name all fields with a user-readable name. Fields that are not named result in a default name of the column number in the result set.
- **Action** procedures should not produce any results, as they are not processed. If an exception is thrown during execution, only the error is logged.

The following is an example stored procedure:

```
CREATE PROCEDURE [cust].[PSD123_InsertedCriticalBehaviorRole_Field]
@IC_EVENT_ACTION_TYPE CHAR(1),
@IC_EVENT_TIMESTAMP DATETIME,
@IC_EVENT_TABLE VARCHAR(128),
@IC_EVENT_COLUMN VARCHAR(128),
@IC_EVENT_KEY_ID INT,
@IC_EVENT_PERSON_ID INT,
@IC_EVENT_CALENDAR_ID INT,
@IC_EVENT_SCHOOL_ID INT,
@IC_EVENT_DISTRICT_ID INT,
@IC_EVENT_DATA VARCHAR(MAX),
@IC_EVENT_USER_ID INT,
@IC_EVENT_TOOL_ID VARCHAR(50)
AS
SELECT TOP 1 r.role, c.[name] calendarName, CONVERT(VARCHAR, i.[timestamp], 101) incidentDate
FROM dbo.BehaviorRole r
INNER JOIN dbo.BehaviorEvent e ON e.eventID = r.eventID
INNER JOIN dbo.BehaviorIncident i ON e.incidentID = i.incidentID
INNER JOIN dbo.Calendar c ON r.calendarID = c.calendarID
WHERE r.resolutionID = @IC_EVENT_KEY_ID
```

Add a Record to the CampusEventProcedure Table

A record should be inserted into the CampusEventProcedure table. This causes the procedure to be available as an option in the correct location in the configuration tools. The CampusEventProcedure has the following columns that should be provided:

Column Name	Description
procedure	Name of the stored procedure as it exists in the database. For example, "PSD123_InsertedCriticalBehaviorRole_Template".
displayName	Name visible in the dropdown list in the configuration tools. For example, "Behavior fields for critical involvements".
hidden	A boolean value that can be used to hide options. Normally this is zero (0) for available stored procedures.
procType	This is used to identify stored procedures as either an action (Action stored procedures), condition (Condition stored procedures), query (Field stored procedures) or userquery (Recipient stored procedures).
schema	This is the name of the schema in which the stored procedure is located. For customer provided procedures, this should be **cust** (while procedures created by Infinite Campus typically use **dbo**).

Best Practices for Stored Procedures

Test all event configurations in a staging or test environment first, even if changes are small .

- Turn on event logging while testing to assist in verifying behavior.
- Be thorough when checking all ways that users may affect the data of interest.
- Be thorough when checking all possible conditions of the data before and after a change that causes an event.

Do not use poor-performing statements in stored procedures .

- This may cause overall application performance issues.
- When defining events, be specific. Use built-in filters for insert/update/delete and specify a column with before/after data values where possible. That allows the application to avoid as much processing work as possible.

Avoid database object name collisions and always specify schema with the schema prefix on object names.

- If a table named *Enrollment* is created in the cust schema, it may cause unusual application issues (and not just in Events & Actions), since there is an enrollment table in the dbo schema. Parts of the application may not correctly qualify which Enrollment table, and instead of using the Campus table located in the dbo schema, it may find the different table in the cust schema.
- Always qualify names with the schema prefix to avoid potential issues. This is especially important for SQL statements in procedures that are themselves not located in the dbo schema.

Don't assume conditions in the database .

- Since processing is done asynchronously after an event has fired, data may have changed in the intervening time frame. An event may fire for a new record of interest, but by the time processing of that event occurs, that record may have been deleted or updated (perhaps it was created by mistake). It is good to confirm important aspects of the data for an event by querying for them as part of the process.
- A JOIN clause or an EXISTS sub-query to the record in question that checks desired conditions avoids unwanted processing or errors in processing.

Make sure rules enforced in Infinite Campus are similarly enforced in your procedures .

- Don't insert invalid values.
- Insert all required fields.
- Don't violate application expected states. Attendance, Roster and Enrollment records must have consistent dates for Infinite Campus to properly function in some areas like Attendance Reports.

Avoid loops and event "amplification" in events.

- If processing one event causes a second event, make sure the second event does not eventually cause the first event again. Loops like this may cause serious issues including eventual application failure.
- Avoid events fired by changes in these tables as that may cause processing loops as well. Normal processing of events can modify event tables and messenger related tables.
- Avoid events that cause multiple secondary events, which may cause multiple further events, etc. This amplification may cause significant performance issues if processing volume is high.