Workflow

Version 4.0.2

Help Guide

June 2022

Anthology Inc.

5201 Congress Avenue
Boca Raton, FL 33487
Main: +1.561.923.2500
Support: +1.800.483.9106
www.anthology.com

CONFIDENTIALITY NOTICE:

The information contained in this document is confidential. It is the property of Anthology Inc. and shall not be used, disclosed, or reproduced without the express written consent of Anthology Inc.

Revision History

Rev.	Date	Description
01	July 2021	Initial release for Workflow Composer Version 4.0. See What's New.
02	Sep. 2021	Added note about CrmConnection string. See What's New.
03	Sep. 2021	Updated note about Password field. See CreatePortalAccount.
04	Oct. 2021	Added ExecuteDataReader Example 3 and ExecuteQuery Example 2. Removed references to V1 activities.
05	Apr. 2022	Release for Workflow Composer Version 4.0.1. See What's New.
06	Jun. 2022	Release for Workflow Composer Version 4.0.2. See What's New.

Contents

Get Started

Welcome to Workflow Help

This help system assists users in recognizing and using the features of workflows and eventing. Use the help system to:

- Learn about the programming concepts related to workflows such as contracts, events, and entities
- Learn how to use the Workflow Composer
- Learn about workflow activities
- Review sample workflows

This help system supports the current Workflow Composer version and two prior versions. Help topics that have been added or modified display a version selector at the top of the topic. Use the version selector to reveal help content associated with prior versions.

Related Help Systems and APIs

https://help.campusmanagement.com/Content/Home.htm

https://www.mycampusinsight.com/Documentation-Center/Help/Help_Home/Content/helphome.htm (logon required). The Object Library for Anthology Student is available under APIs > Anthology Student Object Library.

What's New

Version 4.0.2

- The Windows security message is no longer displayed when installing Workflow Composer using the Click-Once application. See Installation.
- The nlog.config file no longer exists (see NLog). The NLog configuration settings have been placed in the app config file (WorkflowComposer.exe.config).

Version 4.0.1

- Added the option to change the Configuration upon launching Workflow Composer.
- Updated Install Packages (step 4) and added a note below the table in End-of-Life for Anthology Student Activities (V1).
- When a new version Workflow Composer is installed, previously installed packages are automatically uninstalled, and the user is reminded to install packages. See Package Manager and Workflow Composer.

Version 4.0

- Added note about CrmConnection string to ExecuteDataReader, ExecuteNonQuery, and ExecuteQuery.
- **Note**: Forms Builder 3.6 introduces the "CrmConnection" string in the web.config of Forms Renderer (see Renderer Connection Strings). If you have created workflows with What's New activities, ensure that connection strings in the activities match the updated web.config of Forms Renderer.
- Removed V1 activities. See End-of-Life for Anthology Student Activities (V1).
- Added Run Time Messages About V1 Activities.
- Added CRM Configuration option to Workflow Composer. See Configuration.
- Added Authentication for CampusLink API Calls and API Errors with SyRegistry Authentication.
- The status bar of Workflow Composer now displays the installed versions of Anthology Student and Forms Builder. See Workflow Composer.
- Added CommandTimeout property to ExecuteDataReader, ExecuteNonQuery, and ExecuteQuery.
- Added new database event types for Anthology Student. See Anthology Student Database Events.
- Added Use the Http Header for Authentication.
- Added workflow example Create a Student Enrollment Period.
- Added Cloud Subscriptions to the Event Logs topic.

- Added the note about formatting the `<DateTime>` property using the Kendo library. See CreateTask (V2) activity.
- Workflow Composer 4.x requires Microsoft .NET Framework 4.8. For more information, see
  - https://dotnet.microsoft.com/download/dotnet-framework/thank-you/net48-web-installer
  - https://support.microsoft.com/en-us/help/4503548/microsoft-net-framework-4-8-offline-installer-for-windows
- Updated instructions for the ClickOnce installation. See Installation.
- Updated instructions for package installations. See Install Packages.
- Updated Note in Password field of CreatePortalAccount.
- Added ExecuteDataReader Example 3 and ExecuteQuery Example 2.
- Removed references to V1 activities.
- Product renaming and rebranding:
  - CampusNexus Student is now Anthology Student
  - CampusNexus Cloud is now Anthology Cloud.

Overview

Anthology Inc. enables customers to integrate products such as Anthology Student and Forms Builder. Customers can leverage investments made into existing products and at the same time gain immediate value for investments in next generation products that will feature a unified architecture and data model.

An event-driven architecture using tools like Microsoft Visual Workflow integrates existing products with a service bus that customers may have already implemented at their institutions to synchronize data between systems. Workflow empowers users to easily write code to do specific tasks currently not available in existing products or tasks that involve exchanging data between systems. Anthology Web Services are available to facilitate inserting data back into the existing systems.

Scenario
A student updates her phone number in Anthology Student Portal. In the current architecture, the update will be propagated into Anthology Student. With Workflow, the update event can be saved onto the service bus and other database systems deployed at the institution (e.g., CRM[1], LMS[1], and POS[1]) will be updated automatically.

The key objectives of Workflow are:

- Ease of use
- Greater flexibility for the implementation of business processes
- Greater flexibility for the integration with other systems

Workflow uses out-of-the-box .NET functionality such as:

- Security
- Logging and Instrumentation
- Localization/Globalization
- Component Model (Inversion of Control/Dependency Injection Framework)
- Caching

The Event Broker and Workflows components provide the extended business functionality.

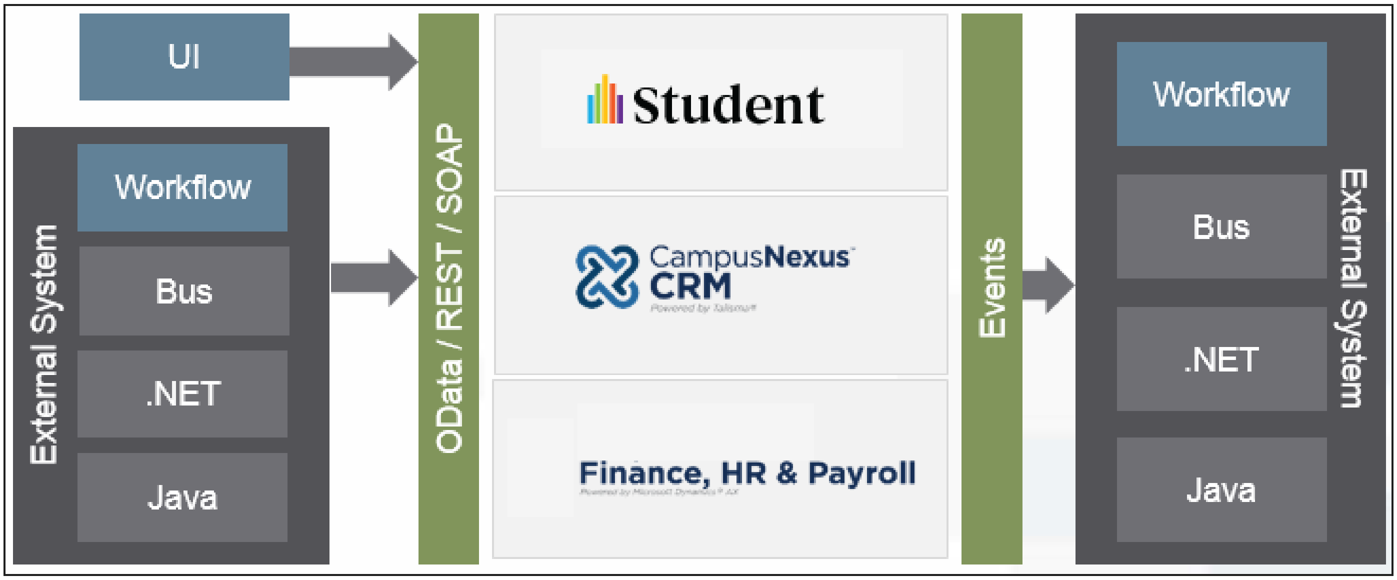
[1] Constituent Management System
[1] Learning Management System
[1] Point of Sales

Event Driven Architecture

Anthology products are based on an event-driven architecture (EDA) in which a software element executes in response to receiving one or more event notifications. The main components in this architecture are the Event Broker and Workflows. Events are utilized in Workflows to perform specific activities in response to the events. Each event can be used to trigger one or more activities.

Event Broker

The Event Broker is a software component that allows different software elements to work together. Service Contracts and Event Contracts constitute the Event Broker.



There is no user interface for the Event Broker. It operates in the background and allows users to focuse on the business logic.

Workflows

Workflows are discrete tasks based on business rules and requirements. Anthology provides workflow activities, that is, 'chunks of code', for power users to compose tasks that are meaningful in a specific environment. Workflows also allow customers to audit or track business processes.

Workflows open the Anthology interfaces to:

- Customers
- Professional Services
- Third party vendors for integration with their systems

You can use Workflow Composer to create workflows. In Workflow Composer, expressions in the Designer must be written in Visual Basic (.NET).

Required Skills

The Workflow Composer application is intended to be used by staff members with the following knowledge and skills.

Prerequisite Knowledge

- Understanding of business processes
- Understanding of Anthology Student application and schema and/or CampusNexus CRM application and schema
- Awareness of .NET technologies and understanding of VB.NET
  - Creating variables, assigning data types, and a basic understanding of development languages
- Awareness of:
  - Windows Workflow Foundation
  - CSS themes
- SQL Knowledge
  - Ability to create SQL jobs, call stored procedures and write queries
- General development knowledge of variables, arguments, control logic, exception handling, debugging, etc.

Advanced Forms Builder and Workflow Development

Expertise in the following is recommended:

- AngularJS (expressions)
- OData
- REST (JSON)
- Bootstrap (themes)
- Workflow tracking and persistence
- TSQL skills to write stored procedures

Security Enhancement for OData Queries

The focus of the security enhancement for OData queries are system integrations with Anthology Student APIs. Integrations use Commands, REST services, and OData endpoints. Previously, all OData endpoints (queries) were available to any authenticated user. If users (even API users) were authenticated, they had full access to all the queries. This enhancement secures access to OData endpoints in the Query Model in the same manner as in the Command Model APIs. If a user, outside of the Anthology Student web client UI, attempts to access a Query Model to which they have no access, the controller will respond with a status "401 Unauthorized".

The OData endpoint security enhancement takes effect with the following releases:

- Anthology Student 21.2.0
- Faculty Workload Management (FWM) 1.2.0
- Financial Aid Automation (FAA) 8.2.0
- Regulatory 12.2.0

OData Query Authorization

In prior versions of these products, once a user (or 3rd party) was authenticated in Anthology Student, all OData endpoints were available for use and all OData queries were available. Access to the Query Model was not restricted via NetSqlAzMan (NSA) in the Security Console.

The OData endpoint security enhancement establishes NSA authorization for the Query Model by adding all Query Model entities to the NSA configuration file. All query operations in NSA are contained in the new Task "All Query Operations" in the Security Console. For backward compatibility, the Task “All Query Operations” has been assigned to the CMC System Administrators Role. This task needs to be added to any other additional roles where backwards compatibility is desired. For the future, individual organizations can create custom tasks from the operations added to the model as needed and assign them to roles as required.

With this enhancement, access to queries is restricted and query operations for each entity are added. Examples of the query operations include:

- Academics.Course.Query
- Common.Student.Query
- Crm.DocumentType.Query

The naming pattern for query operations is: **`<Module>.<Entity>.Query`**

Users executing OData queries will either need a QueryToken (cookie) provided by the Anthology Student web client UI or authorization granted in NSA for specific Query Model entities requested in the query.

Note: Users logging in via the web client for Anthology Student will not be affected by this change. Access to the various areas of the application continues to be controlled via the Tasks assigned in the Security Console.

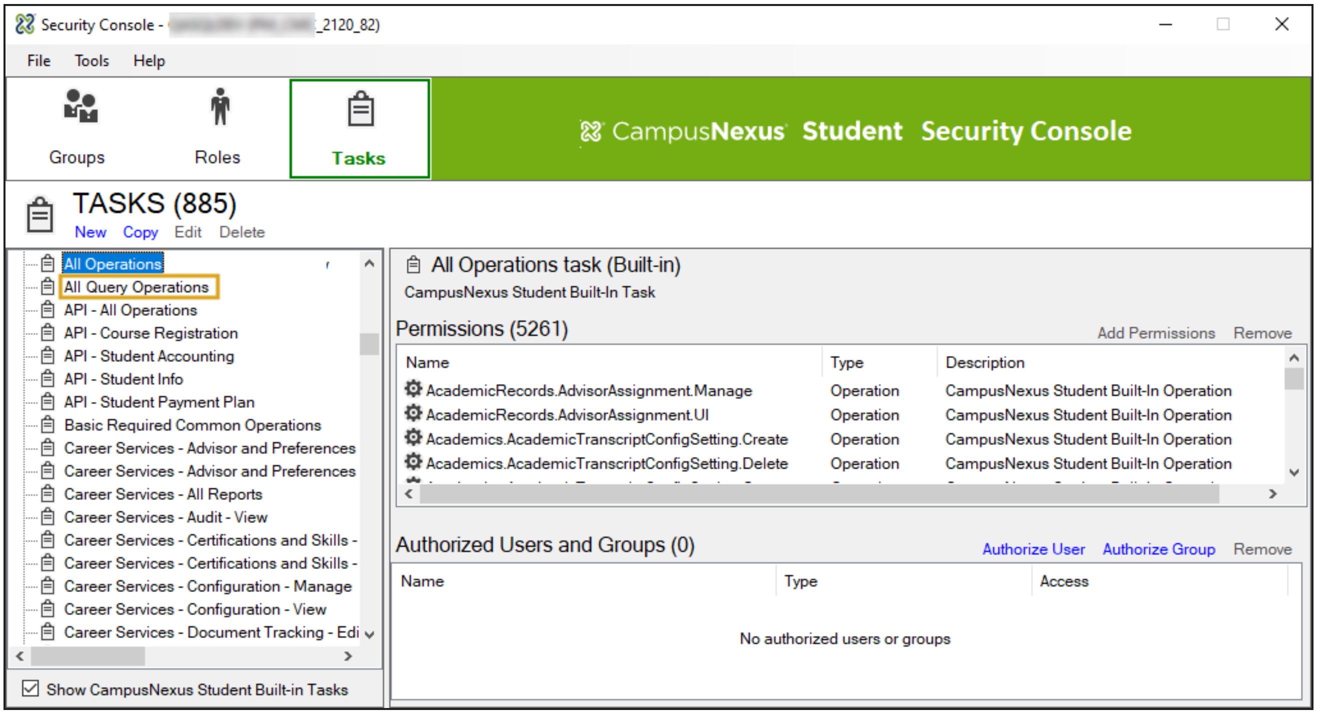
The enhancement requiring OData query authorization may impact the following audiences:

- Partners doing integrations with Anthology Student
- Clients who have already leveraged this ability in prior versions of the product
- Client implementations that use custom logic created by our Professional Services team
- Professional Services teams working on integration projects

Configure OData Query Authorization

When you begin working with Anthology Student version 21.2 or any of the other product versions above, you need to go into your Security Console and either:

- Grant everyone who needs access to this capability the Task **All Query Operations**. This is not the recommended approach but mirrors existing functionality.


Security Console - [redacted]_2120_82)
File
Tools
Help
Groups
Roles
Tasks
CampusNexus Student Security Console
TASKS (885)
New
Copy
Edit
Delete
All Operations
All Query Operations
API - All Operations
API - Course Registration
API - Student Accounting
API - Student Info
API - Student Payment Plan
Basic Required Common Operations
Career Services - Advisor and Preferences
Career Services - Advisor and Preferences
Career Services - All Reports
Career Services - Audit - View
Career Services - Certifications and Skills -
Career Services - Certifications and Skills -
Career Services - Configuration - Manage
Career Services - Configuration - View
Career Services - Document Tracking - Edi
☑ Show CampusNexus Student Built-in Tasks
All Operations task (Built-in)
CampusNexus Student Built-In Task
Permissions (5261)
Add Permissions
Remove
Authorized Users and Groups (0)
Authorize User
Authorize Group
Remove

Name	Type	Description
AcademicRecords.AdvisorAssignment.Manage	Operation	CampusNexus Student Built-In Operation
AcademicRecords.AdvisorAssignment.UI	Operation	CampusNexus Student Built-In Operation
Academics.AcademicTranscriptConfigSetting.Create	Operation	CampusNexus Student Built-In Operation
Academics.AcademicTranscriptConfigSetting.Delete	Operation	CampusNexus Student Built-In Operation

Name	Type	Access

No authorized users or groups

Note: The "All Query Operations" Task is not assigned to the Cmc System Administrators Role. The administrator Role (Group) already has an “All Operations” Task that includes the new query operations. This is done automatically.

— OR —

- Build **custom tasks** for groups/roles and grant them access to the query operations they need. Query operations would then be added to Tasks as necessary. You can filter the operations and entities to create custom tasks. This is the recommended long term approach.

One Task which includes all of the Operations could be added, for example, "System.Query.All" or similar. This would enable the same behavior that is currently provided for users of the Anthology Student web client UI, who currently have access to all entries in the OData Query Model.

Workflows and OData Query Authorization

Activities in Workflow Composer that use OData endpoints will fail if the **APIUser** does not have the necessary OData authorization. By default, the APIUser is a member of the Administrator group which has the OData authorization. However, if a client has a custom APIUser, this user will need access to the OData query operations.

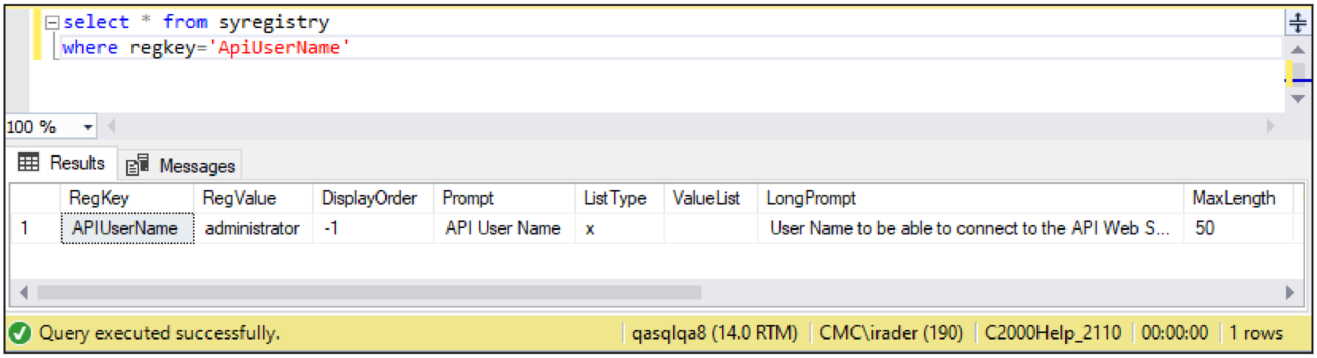
All workflow activities with properties that are populated by OData queries will require OData query authorizations.

Note: The ExecuteODataQuery<> activity will fail with a "401 Unauthorized" response if the APIUser does not have access to any of the entities referenced in the query. If multiple entities are included in an OData query, the user must be authorized to ALL of the entities in order for the query to execute.

Example

A user adds a “Create Task” activity to a workflow. This activity has drop-down list for Task Template and Task Status which are populated by OData calls.

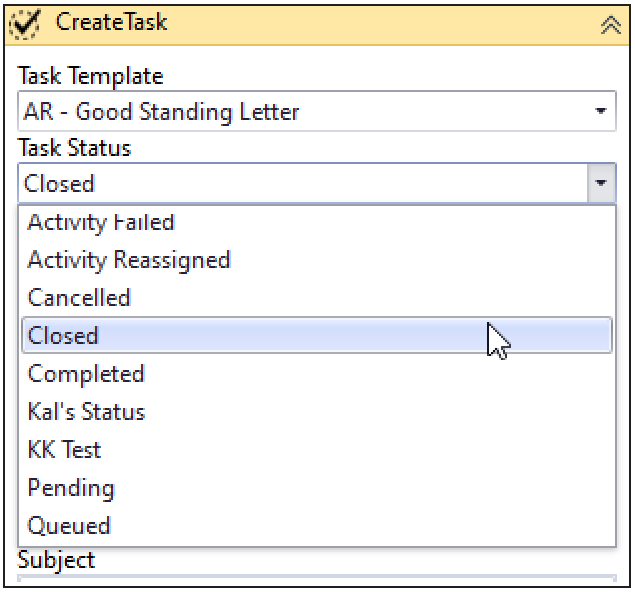
1. When the user's APIUserName in the SyRegistry table is "administrator", the drop-down lists are populated without error because as the"“administrator" user is part of the Administrators Group.



```
select * from syregistry
where regkey='ApiUserName'
```

	RegKey	RegValue	DisplayOrder	Prompt	ListType	ValueList	LongPrompt	MaxLength
1	APIUserName	administrator	-1	API User Name	x		User Name to be able to connect to the API Web S...	50

Query executed successfully. | qasqlqa8 (14.0 RTM) | CMC\irader (190) | C2000Help_2110 | 00:00:00 | 1 rows



Task Template

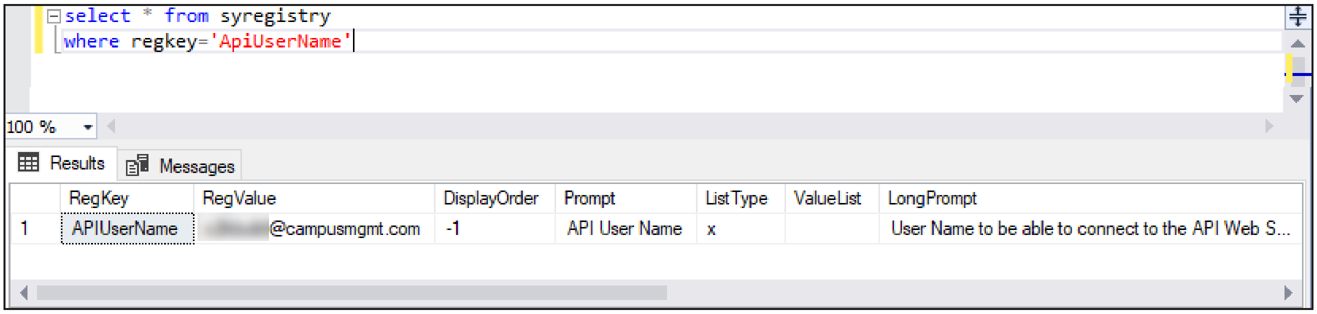
AR - Good Standing Letter

Task Status

Closed

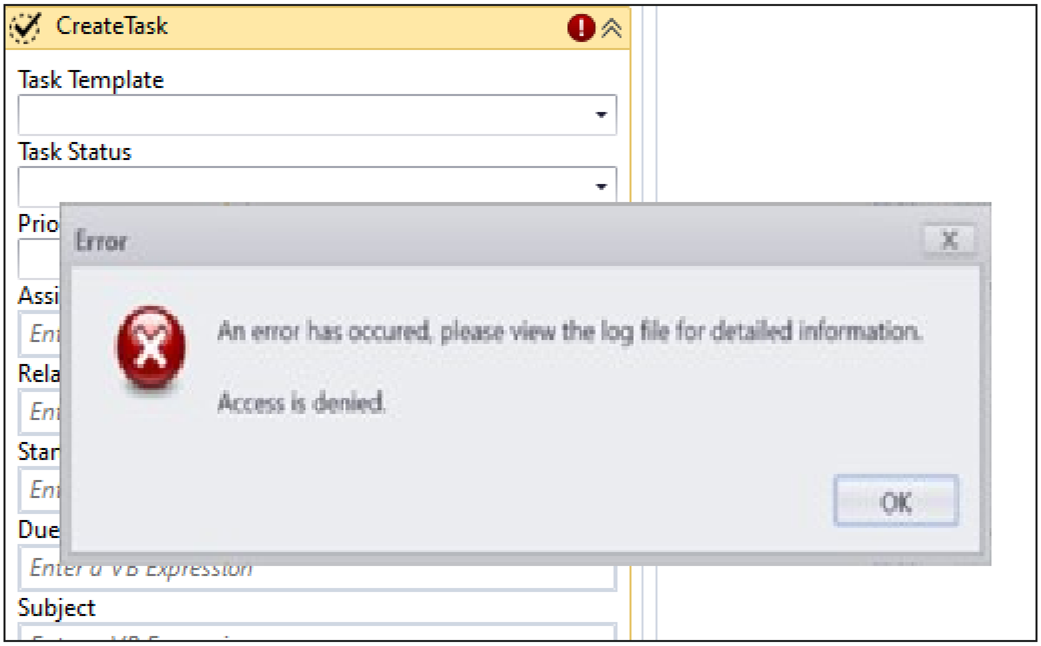
Subject

2. When the APIUserName is a different account (user@campusmgmt.com), the CreateTask activity fails with an "Access denied" error.



```
select * from syregistry
where regkey='ApiUserName'
```

	RegKey	RegValue	DisplayOrder	Prompt	ListType	ValueList	LongPrompt
1	APIUserName	[redacted]@campusmgmt.com	-1	API User Name	x		User Name to be able to connect to the API Web S...

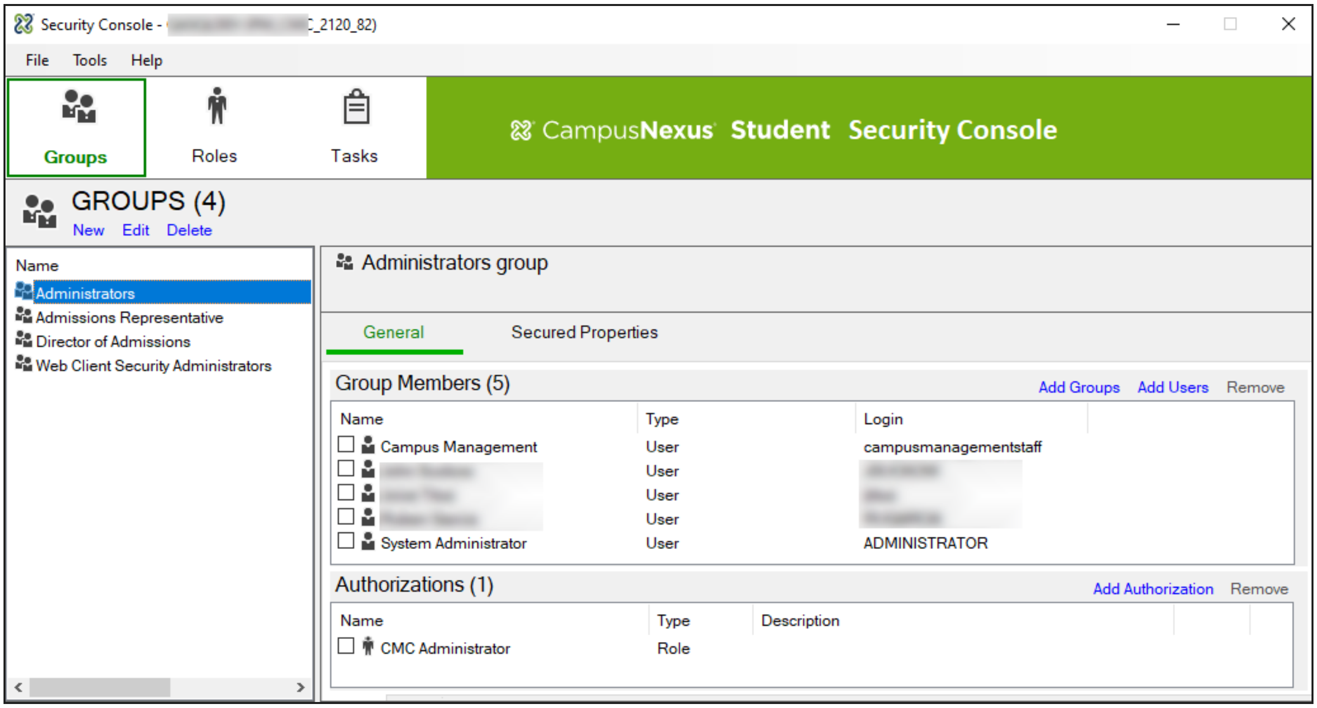


An error has occured, please view the log file for detailed information.

Access is denied.

3. After adding the account (user@campusmgmt.com) to the Administrators Group in the Security Console and clearing the cache in the UI to remove the cached NSA items, the CreateTask activity works again.

   Instead of adding the user account to the Administrators Group, the user account could have been granted specific access to each OData query operation.



Security Console - [redacted]_2120_82)

GROUPS (4)

Name
Administrators
Admissions Representative
Director of Admissions
Web Client Security Administrators

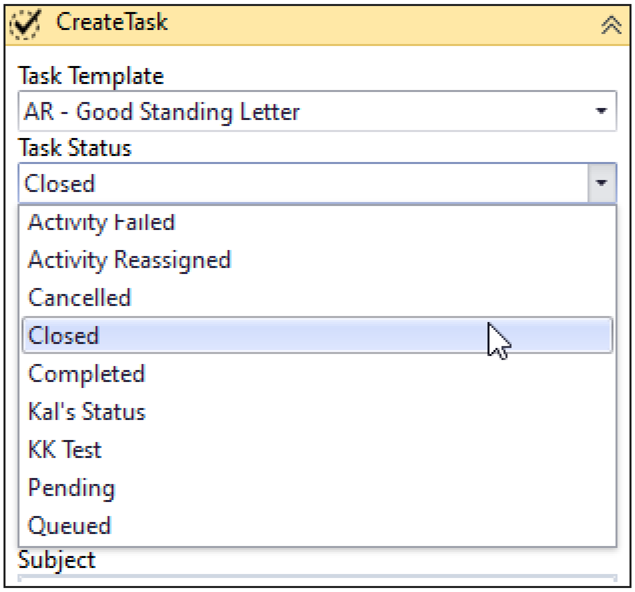
Administrators group

Group Members (5)

Name	Type	Login
☐ Campus Management	User	campusmanagementstaff
☐ [redacted]	User	[redacted]
☐ [redacted]	User	[redacted]
☐ [redacted]	User	[redacted]
☐ System Administrator	User	ADMINISTRATOR

Authorizations (1)

Name	Type	Description
☐ CMC Administrator	Role	



CreateTask

Task Template: AR - Good Standing Letter

Task Status: Closed

- Activity Failed
- Activity Reassigned
- Cancelled
- Closed
- Completed
- Kal's Status
- KK Test
- Pending
- Queued

Subject

Workflow Composer

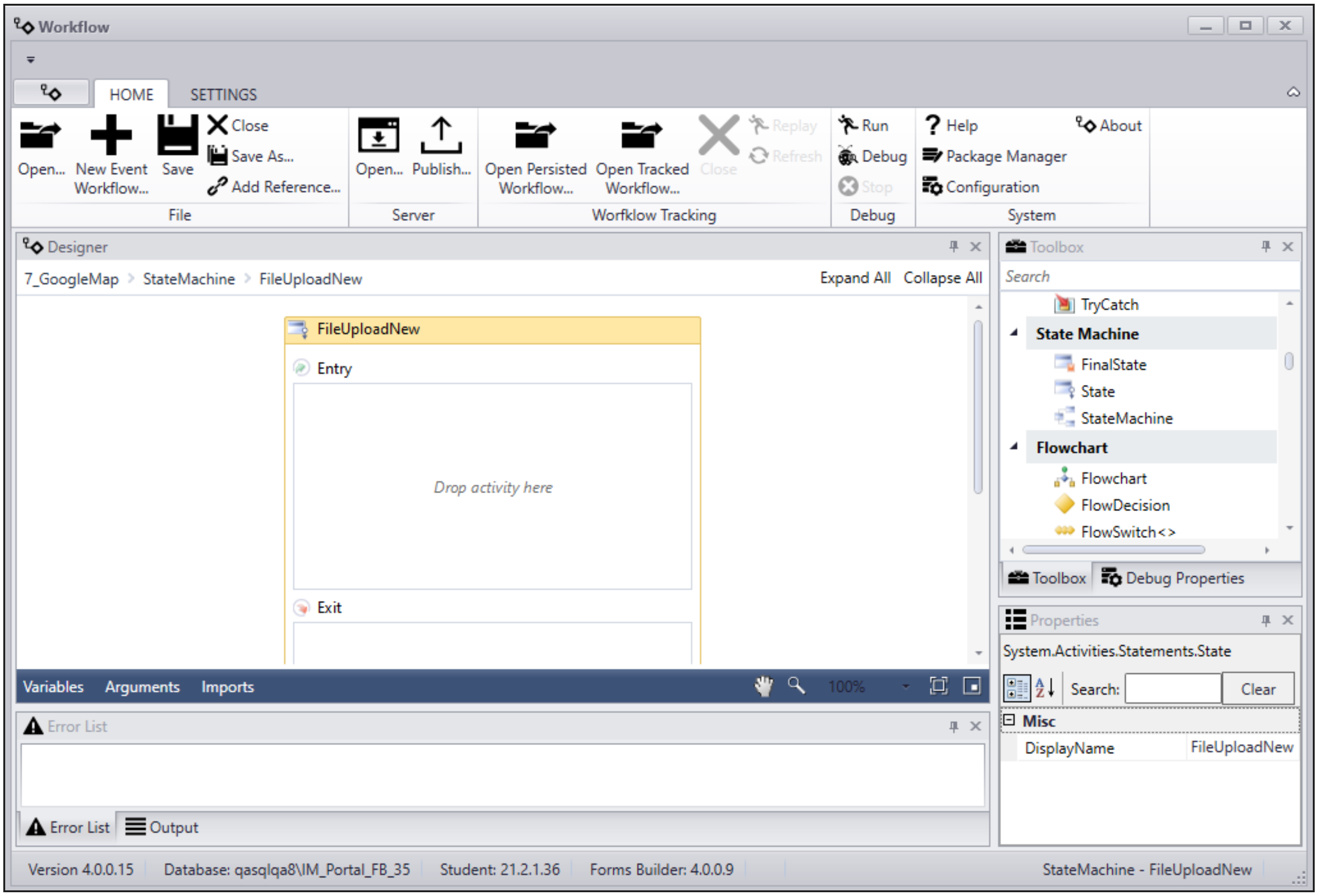
Workflow Composer UI

This topic describes the user interface (UI) of Workflow Composer. Some UI elements are visible when the application is opened, while additional UI elements become available when a workflow file is loaded.

When the application is opened for the first time, the UI consists of a ribbon and several task panes or windows. The ribbon organizes commands into logical groups. These groups appear on separate tabs in a strip across the top of the window. The task panes include the Designer area where the workflow sequence is composed and several resource panes.

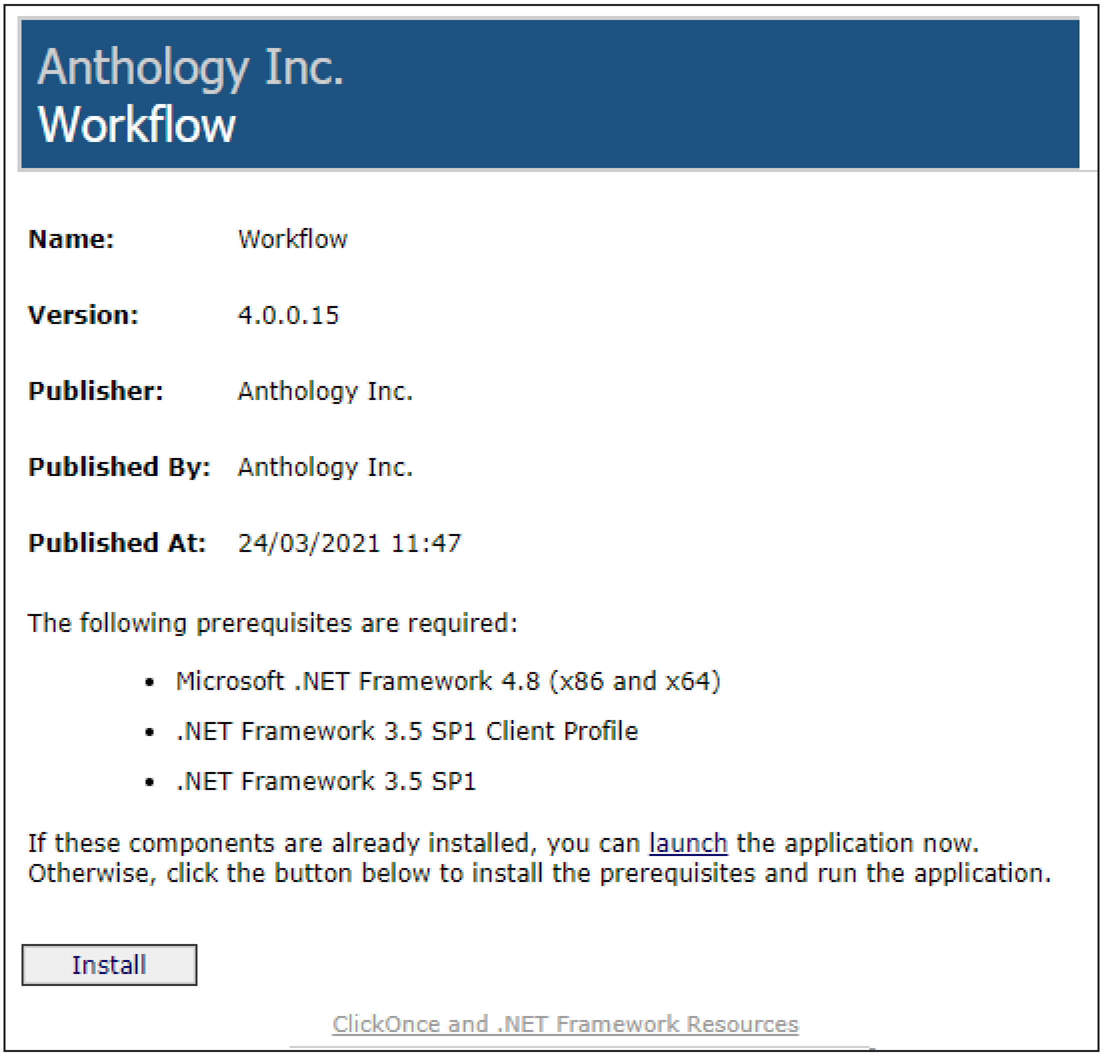
The status bar displays the following:

- Version of Workflow Composer
- Name of the connected database
- Version of Anthology Student
- Version of Forms Builder
- Name of the activity selected in the Designer pane



Installation

Workflow Composer is deployed via a **ClickOnce** application that allows self-updating Windows-based applications to be installed and run with minimal user interaction. You can install Workflow Composer with one click on the **Install** button or **launch** it from a web site.



Anthology Inc.
Workflow

Name: Workflow

Version: 4.0.0.15

Publisher: Anthology Inc.

Published By: Anthology Inc.

Published At: 24/03/2021 11:47

The following prerequisites are required:

- Microsoft .NET Framework 4.8 (x86 and x64)
- .NET Framework 3.5 SP1 Client Profile
- .NET Framework 3.5 SP1

If these components are already installed, you can launch the application now.
Otherwise, click the button below to install the prerequisites and run the application.

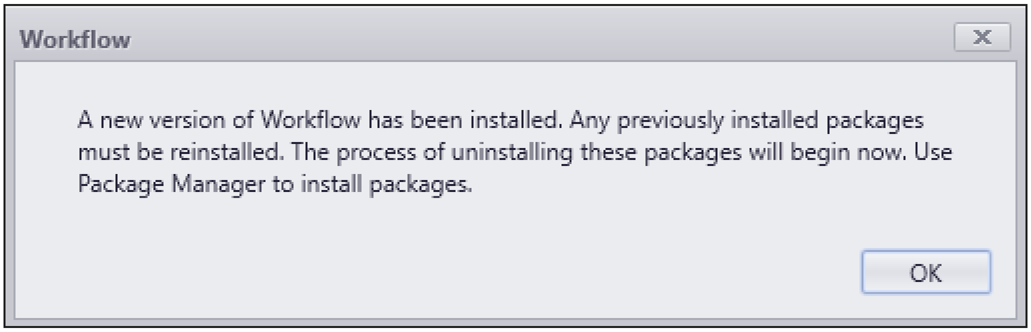
⚠ Depending on the settings and antivirus/malware software installed on your machine as well as your corporate policies, you may see a warning when installing Workflow Composer and its activity packages.

When the installation is completed, you are prompted to configure Workflow Composer. For more information, see Configuration.

For details about the ClickOnce URL and login credentials, refer to **https://filetransfer.campusmgmt.com** > **softwareupdates** > **WorkflowComposer** > **WF_ComposerInstallationSteps.pdf**.

Note: If you installed Workflow Composer using ClickOnce with auto update, previously installed packages are removed and need to be reinstalled.

When a new version of Workflow Composer has been installed, the following message will remind you to reinstall any packages.



A new version of Workflow has been installed. Any previously installed packages must be reinstalled. The process of uninstalling these packages will begin now. Use Package Manager to install packages.

Click **OK** and proceed to install the needed Activities and Contracts .msi packages using Package Manager.

For each .msi package that you install, you will be prompted to confirm that you want to allow the app to make changes to your device.

Tip: Once the automatic first launch with package cleanup is completed, right-click the Workflow Composer icon and select “Pin to taskbar” and/or keep a shortcut on your desktop. That way you can easily find the install location again and the latest version of the app is used each time.

Ribbon

The button displays a basic menu that lets you to create, open, and close a workflow, access this Help system, Package Manager, or the About window.

The **About** window displays the following information:

- Version
- Database (name of Anthology Student or CampusNexus CRM database)
- Tracking Database
- Build Date
- Copyright

The ⌃ or ⌄ buttons on the top right show or hide the ribbon.

The **Settings** tab lets you reset the default layout of the task panes or select a color scheme (theme) for Workflow Composer.

Task Panes

The task panes include the following:

- Designer
- Toolbox
- Debug Properties
- Properties
- Error List
- Output

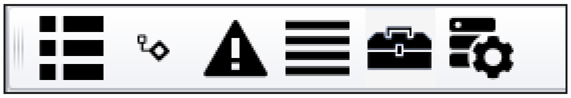
You can customize layout of the panes as follows:

- Move panes to different positions within the main window.
- Detach panes from the main window.
- Re-size panes.
- Hide panes.
- Close panes.
- Re-open panes.

Right-click the title bar of a pane and select from the following display settings for the current session:

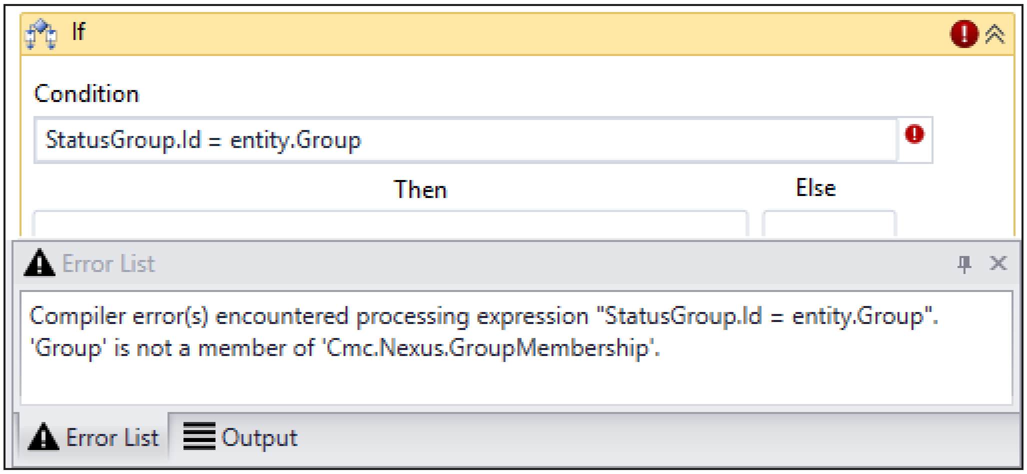
- Float
- Auto Hide
- Hide

When you have closed task panes, icons representing the panes appear at the bottom of the main window. Hover over the icons to see the labels. Click an icon to re-open the associated pane.



Error List and Output Tabs

You can select to view the Error List or the Output tabs below the Designer pane. The Error List helps to identify errors that may occur while building workflows in the Designer pane. For example, if an incorrect entity is used in an expression, an error similar to the example below is displayed.



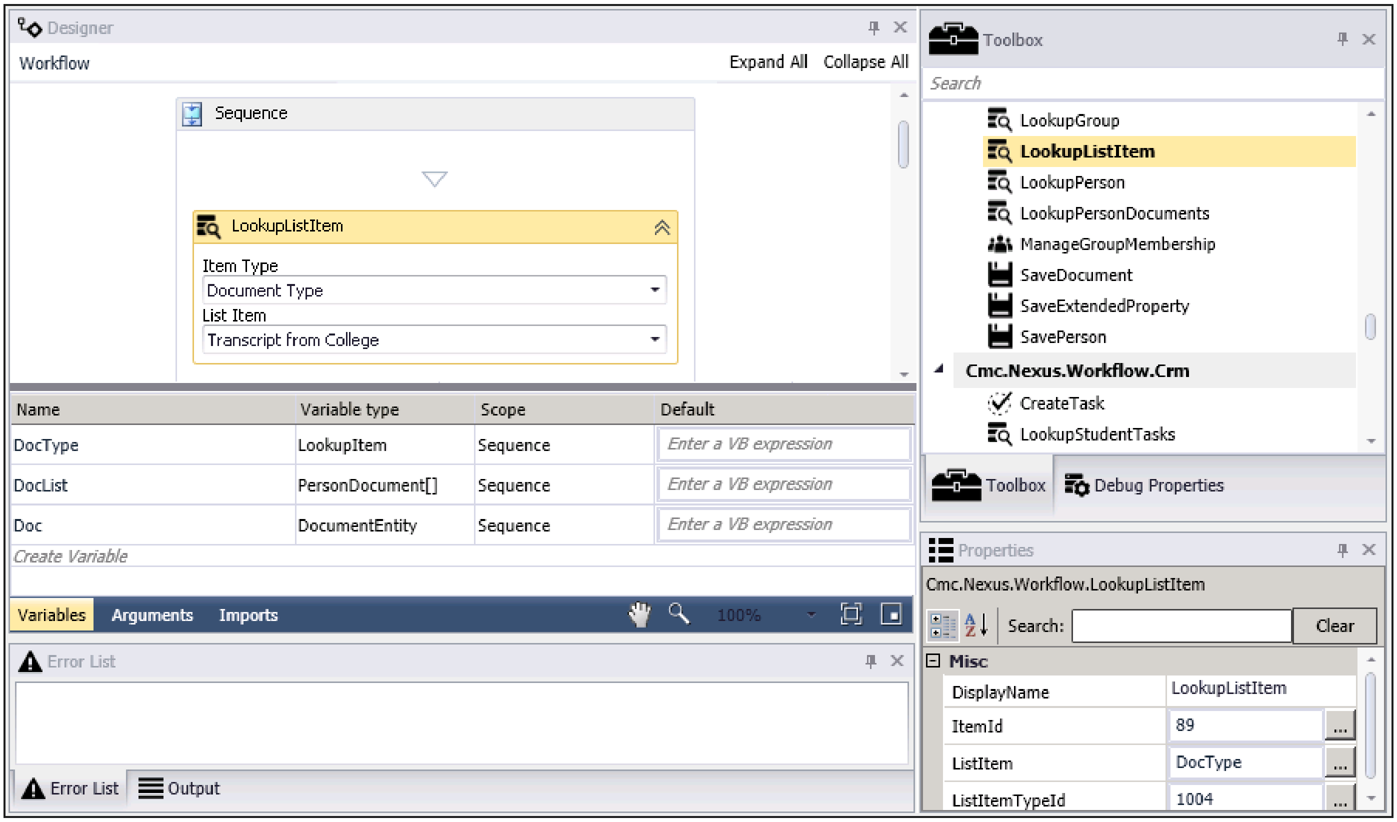
Compiler error(s) encountered processing expression "StatusGroup.Id = entity.Group".
'Group' is not a member of 'Cmc.Nexus.GroupMembership'.

The Error List also indicates any problems encountered with data types for variables. For every activity that requires a variable, an error is displayed until the correct variable is added to the workflow.

Additional UI Elements When a Workflow is Loaded

When a workflow file (.XAML) is loaded into the Workflow Composer, toolbars appear at the top and bottom of the Designer pane.

- The toolbar at the top of the Designer pane displays **breadcrumbs** for the workflow elements, an **Expand All** button, and a **Collapse All** button. The breadcrumbs appear when you double-click the icon in the header bar of a workflow activity.
- The toolbar below the Designer pane displays buttons for **Variables**, **Arguments**, **Imports**, and **pan/-zoom controls**.



Item Type
Document Type
List Item
Transcript from College

Name	Variable type	Scope	Default
DocType	LookupItem	Sequence	*Enter a VB expression*
DocList	PersonDocument[]	Sequence	*Enter a VB expression*
Doc	DocumentEntity	Sequence	*Enter a VB expression*
Create Variable			

Cmc.Nexus.Workflow.LookupListItem

DisplayName	LookupListItem
ItemId	89
ListItem	DocType
ListItemTypeId	1004

Click **Variables** to view, edit, or create variables to be used in the workflow. The variable details include:

- Name
- Variable type
- Scope
- Default

You have the option to create variables in this pane.

Click **Arguments** to view, edit, or create arguments to be used in the workflow. The argument details include:

- Name
- Direction
- Argument type
- Default value

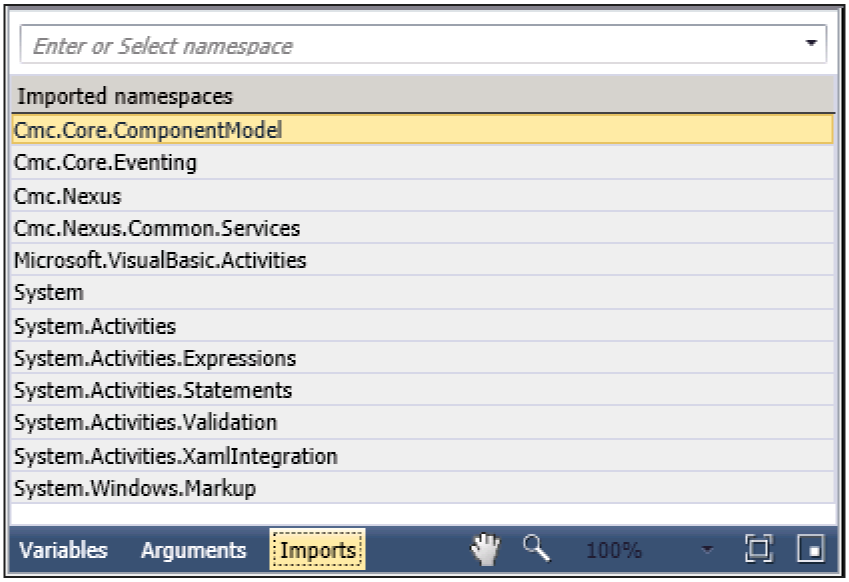
Click **Imports** to view the list of the imported namespaces. The default namespaces include:

- Cmc.Core.ComponentModel
- Cmc.Core.EntityModel

- Cmc.Nexus (multiple namespaces depending on the activities used in the workflow)
- Microsoft.VisualBasic.Activities
- System.Activities (multiple namespaces)
- System.Windows.Markup

You have the option to enter or select additional namespaces for import.

Note: If you copy and paste a Sequence from one workflow to another, you may need to recreate any associated variables to ensure all namespaces are properly imported.



Imported namespaces
Cmc.Core.ComponentModel
Cmc.Core.Eventing
Cmc.Nexus
Cmc.Nexus.Common.Services
Microsoft.VisualBasic.Activities
System
System.Activities
System.Activities.Expressions
System.Activities.Statements
System.Activities.Validation
System.Activities.XamlIntegration
System.Windows.Markup

The **pan/zoom controls** enable you to pan and zoom the display in the Designer pane. Tooltips are provided for these buttons.

Audits

The database for Workflow Composer 3.0.1.8 and later provides a **WorkflowDefinitionVersion_Audit** table that logs workflow definition version changes. Records are inserted into the new table when workflow definition versions are updated (enabled/disabled) and deleted.

Newly published (inserted) workflow definition versions are stored in the **WorkflowDefinitionVersion** table. When a workflow definition version is first published (inserted) no audit records appear in the WorkflowDefinitionVersion_Audit table. However, a **View** of the **WorkflowDefinitionVersion_Audit** provides details of newly created workflow definition versions.

⚠ Workflow auditing is not supported for standalone CampusNexus CRM deployments.

Queries

To query the **View** of the **WorkflowDefinitionVersion_Audit** in MS SQL Server Management Studio, you can use Select statements such as:

select top 10 * from Vw_WorkflowDefinitionVersion_Audit order by DateLstMod desc

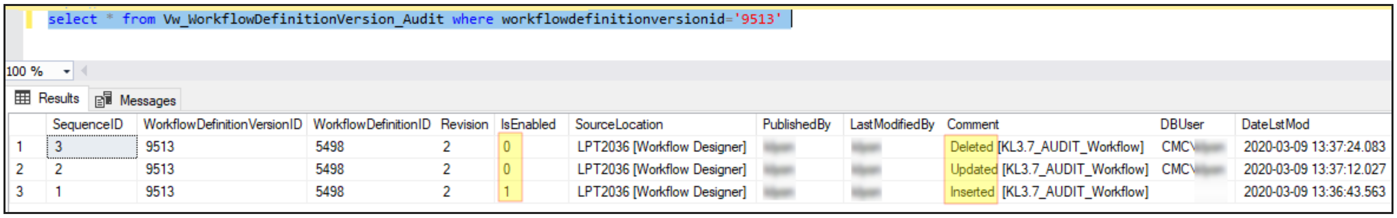
select * from Vw_WorkflowDefinitionVersion_Audit order by Comment

select * from Vw_WorkflowDefinitionVersion_Audit where WorkflowDefinitionVersionID='nnnn'

Examples

To audit the status of your workflow versions, check the values in the **IsEnabled** and **Comment** columns of the WorkflowDefinitionVersion_Audit view.

- When you create, enable, and publish a workflow version, IsEnabled is set to **1**, and the Comment field shows a record is **inserted**.
- When you disable a workflow version, IsEnabled is set to **0**, and the Comment field shows that the record is **updated**.
- When you delete a workflow version, IsEnabled is set to **0**, and the Comment field shows that the record is **deleted**.



```
select * from Vw_WorkflowDefinitionVersion_Audit where workflowdefinitionversionid='9513'
```

	SequenceID	WorkflowDefinitionVersionID	WorkflowDefinitionID	Revision	IsEnabled	SourceLocation	PublishedBy	LastModifiedBy	Comment	DBUser	DateLstMod
1	3	9513	5498	2	0	LPT2036 [Workflow Designer]	[redacted]	[redacted]	Deleted [KL3.7_AUDIT_Workflow]	CMC\[redacted]	2020-03-09 13:37:24.083
2	2	9513	5498	2	0	LPT2036 [Workflow Designer]	[redacted]	[redacted]	Updated [KL3.7_AUDIT_Workflow]	CMC\[redacted]	2020-03-09 13:37:12.027
3	1	9513	5498	2	1	LPT2036 [Workflow Designer]	[redacted]	[redacted]	Inserted [KL3.7_AUDIT_Workflow]	[redacted]	2020-03-09 13:36:43.563

Coding for Activity Errors

To help troubleshoot workflow errors, we recommend that you wrap Anthology activities in a TryCatch activity and use the ValidationMessageCollection property wherever it is available.

ValidationMessageCollection

Almost all Anthology activities provide the ValidationMessageCollection property. This property is designed to detect and log .NET framework and WCF service call exceptions as well as parameter validation exceptions.

ValidationMessageCollection provides built-in arguments.

- In Forms Builder workflows, the argument to use is:

  ```
  formInstance.validationMessages
  ```

- In eventing workflow for Anthology Student or CampusNexus CRM, the argument to use is:

  ```
  args.validationMessages
  ```

In eventing workflows you can also specify the variable of type "Cmc.Core.Eventing.ValidationMessageCollection" (see Capture Validation Errors).

ValidationMessageCollection does not need to be newed up (i.e., a new ValidationMessageCollection is not needed for the Default value). The property value will only be newed up if it is null; otherwise is it appended to previous captured validation messages.

TryCatch

Anthology activities should be wrapped in a TryCatch activity to handle exceptions that are raised at run time. This applies primarily to activities that write to the database (i.e., Save and Update activities). Lookup and Create activities do not need to be embedded in a TryCatch activity.

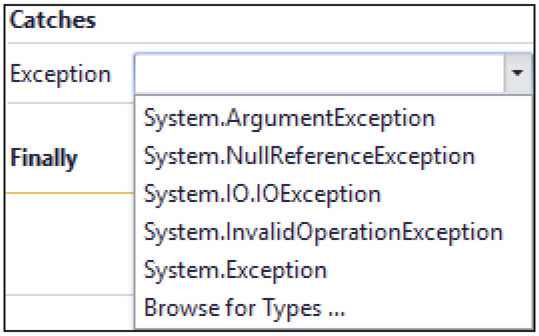
The TryCatch workflow activity has three sections: Try, Catches, and Finally.

Try

Place the Anthology activity for which you want to provide error handling in the Try section. Our example uses a ConvertApplicantToEnrollment activity. The Try section successfully completes if no exceptions are thrown from it.

Catches

Select the exception type in the Catches section. In our example the type is **System.Exception**. You can add multiple catches where each catch handles a different exception type. System.Exception is the catch-all exception and should always be the last exception in the list if you want to trap specific exceptions, otherwise more specific exceptions will never be caught. Catches cannot be reordered. They must be deleted and added in the correct order.



After selecting the exception type, you can add an activity to the catch. In our example a **WriteLine** activity writes exception messages to the console.

```
"Exception: " & exception.Message
```

Note: WriteLine activities are useful when testing workflows with the Run option. Otherwise, use LogLine activities with Level=Error.

The Catches section successfully completes if no exceptions are thrown from it.

Finally

The Finally section includes a Condition that checks if the ValidationMessageCollection has errors. The Condition in our example uses a variable named "valMsgColl" of type "Cmc.Core.Eventing.ValidationMessageCollection".

If an error is found, a **WriteLine** activity writes the text "Validation messages" to the console.

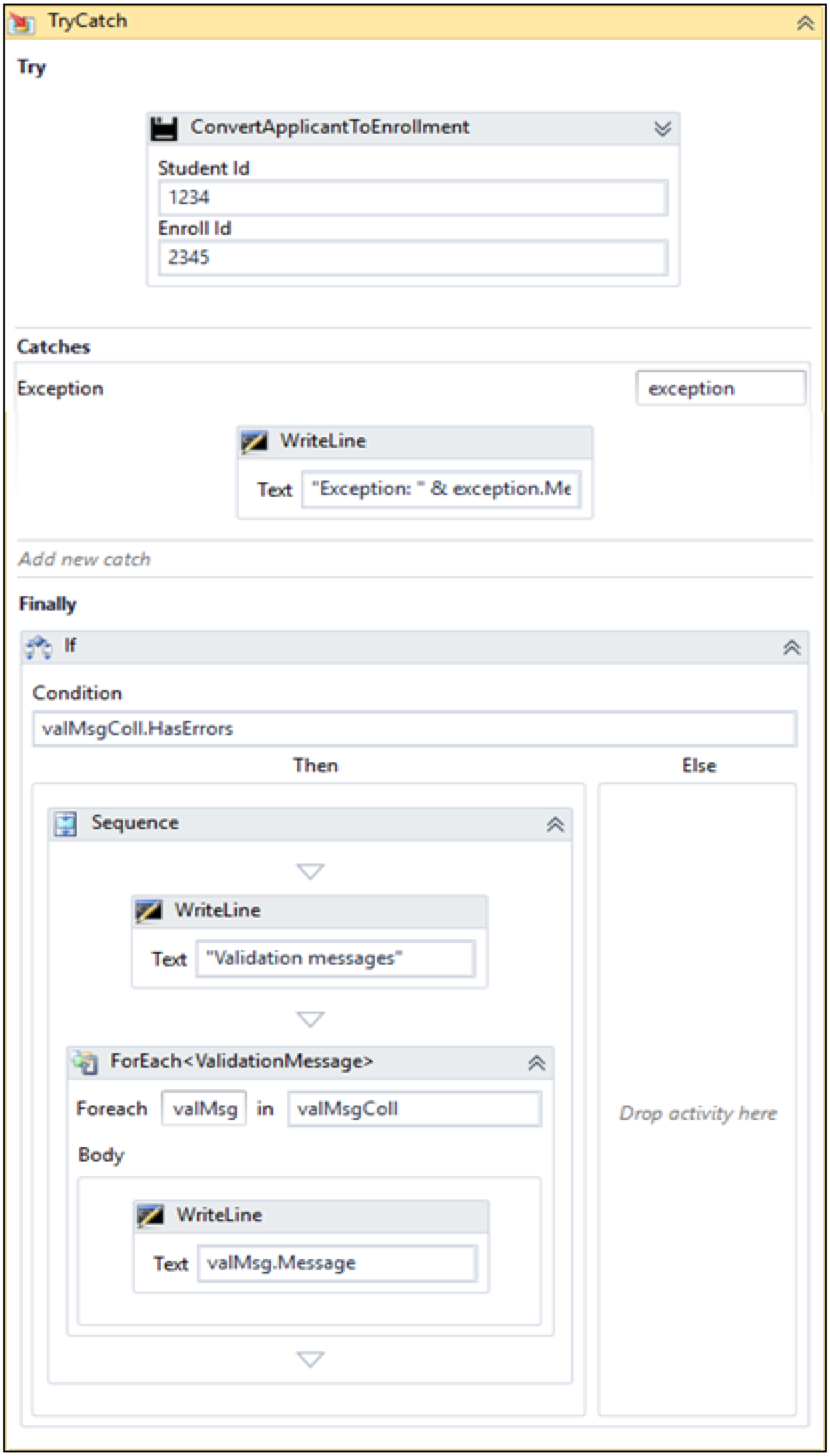
The **ForEach** activity ensures that invalid values in any field of the ConvertApplicantToEnrollment activity will result in a validation message, e.g.:

```
Validation messages
Student Id is not valid

Validation messages
Invalid Academic Advisor selected
```

The console will also display a message if an exception is caught, e.g.:

```
Validation messages
Validation Failed: Field: ProgramVersionId generated an exception during validation.
The following errors were encountered while processing the workflow tree:
'DynamicActivity': The private implementation of activity '1: DynamicActivity' has the following
validation error:   Value for a required activity argument 'GradeLevelId' was not supplied...
```

The activities in the Finally section are executed when either the Try section or the Catches section successfully completes.



TryCatch

Try

ConvertApplicantToEnrollment

Student Id: 1234

Enroll Id: 2345

Catches

Exception: exception

WriteLine

Text: "Exception: " & exception.Me

Add new catch

Finally

If

Condition: valMsgColl.HasErrors

Then

Sequence

WriteLine

Text: "Validation messages"

ForEach<ValidationMessage>

Foreach valMsg in valMsgColl

Body

WriteLine

Text: valMsg.Message

Else

Drop activity here

For more information, see:

- https://docs.microsoft.com/en-us/dotnet/framework/windows-workflow-foundation/exceptions.
- https://docs.microsoft.com/en-us/visualstudio/workflow-designer/trycatch-activity-designer?view=vs-2019

Configuration

For details about the installation of Workflow Composer, please refer to Installation Manager Help.

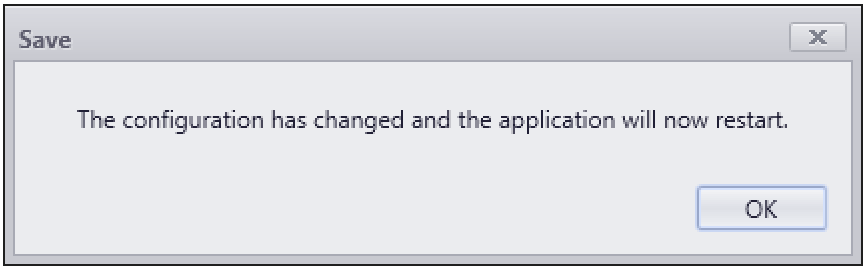
Once Workflow Composer 4.x is installed, you need to specify whether it accesses the databases via direct connections or via a Workflow Web API.

- In an Anthology Cloud 2.0 environment, configure the Workflow Web API Connection. The Workflow Web API replaces the Citrix connections used previously in cloud environments.
- In on-premise or Azure (non-Anthology Cloud 2.0) environments, configure Direct Database Connections.

The configuration needs to be done only once when Workflow Composer is installed the first time. The settings are retained during upgrades.

Workflow Composer 4.0.1 introduces the option to change the configuration upon launching the application. You can choose to **Connect** to the previously configured environment or change the configuration to access a different environment.

If you change the configuration, the following message appears. When you click **OK**, Workflow Composer will restart and connect to the newly configured host. It will take a few seconds to start.



The configuration has changed and the application will now restart.

On initial start up, the Configuration window displays **Connect** and **Exit** buttons. When you select Connect after you have configured the connection, Workflow Composer will launch and connect to the configured host. If you select Exit, Workflow Composer will not launch.

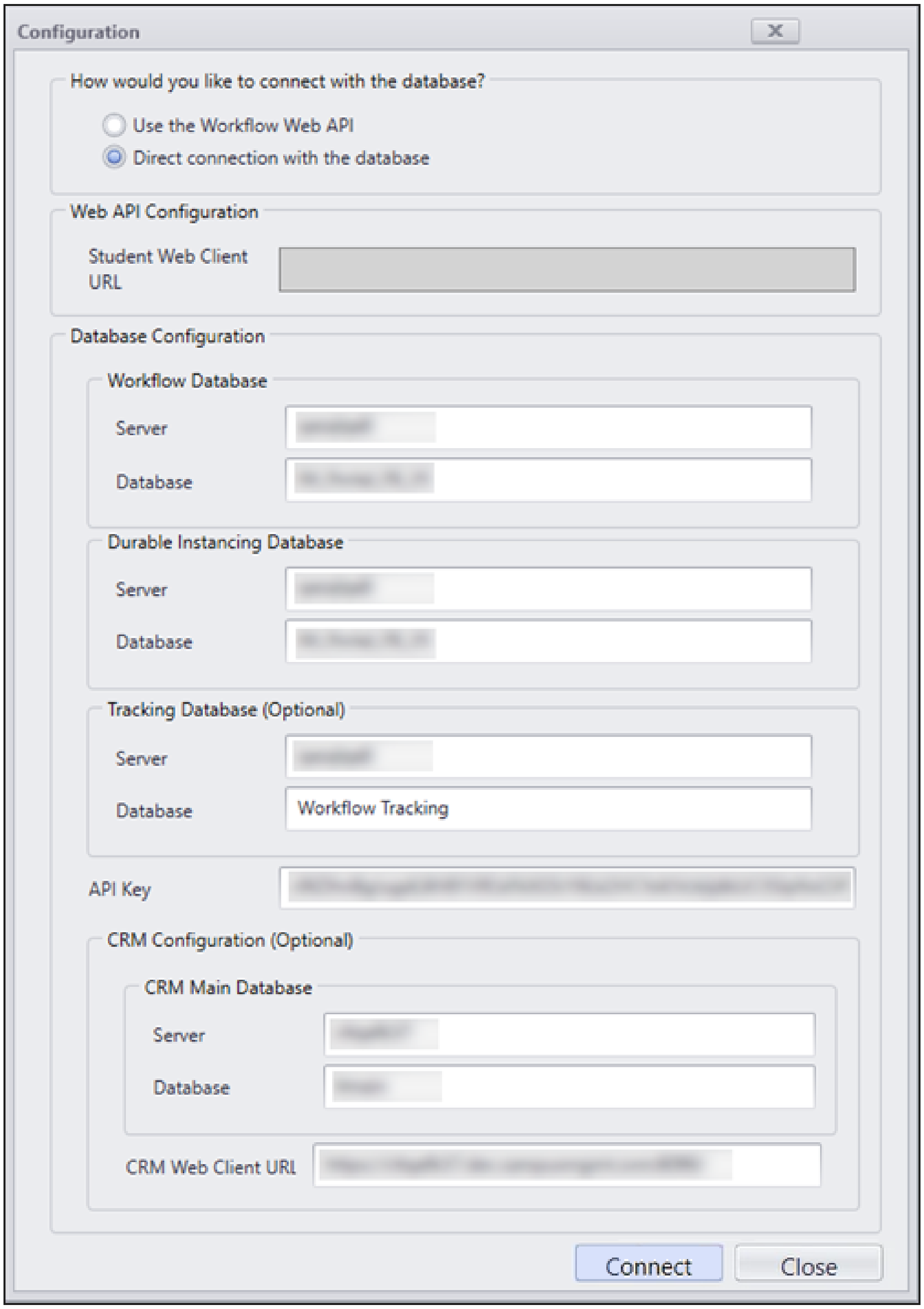
Once Workflow Composer has been launched, the Configuration window displays **Connect** and **Close** buttons. You can update the existing configuration and select Connect to continue working with the updated configuration. If you click Close, the Configuration window will close.

After you click the X button at the top of the Configuration window, you can open the Configuration window again and change the configuration details if needed.

Direct Database Connections

If you are using Workflow Composer with on-premises databases connections:

1. Select **Direct connection with the database**.



Configuration

How would you like to connect with the database?

○ Use the Workflow Web API
● Direct connection with the database

Web API Configuration

Student Web Client URL:

Database Configuration

Workflow Database
Server: [redacted]
Database: [redacted]

Durable Instancing Database
Server: [redacted]
Database: [redacted]

Tracking Database (Optional)
Server: [redacted]
Database: Workflow Tracking

API Key: [redacted]

CRM Configuration (Optional)

CRM Main Database
Server: [redacted]
Database: [redacted]

CRM Web Client URL: [redacted]

2. Specify the server names and database names for your database connections.

- The **Workflow Database** is the database that supplies values to your workflow activities. It can be an Anthology Student or CampusNexus CRM database.
- The **Durable Instancing Database** typically uses the same server and database as the Workflow Database.
- (Optional) The **Tracking Database** is named "WorkflowTracking" by default. It can be on the same server as the Workflow Database and the Durable Instancing Database.

3. In the **API Key** field, specify the key you use to access Anthology Activities and Contracts packages.
4. (Optional) In the **CRM Configuration** section, specify the following:
   - CRM Main Database **Server** and **Database**
   - **CRM Web Client URL**

   By default, the CRM Configuration section will be blank.

   Integrated security will be used for the connection information.

   When the Server and Database are populated under the CRM Main Database, a connection string will be created in the Workflow configuration file named CRMdbconnection.

```
  <connectionStrings>
    <clear />
    <add name="LocalSqlServer" connectionString="data source-
e=.\SQLEXPRESS;Integrated Secur-
ity=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;User Instance=true"
      providerName="System.Data.SqlClient" />
    <add name="dbConnection" connectionString="Data Source=<server>;Initial Cata-
log=<database>;Integrated Security=True;Application Name=&quot;Workflow Com-
poser&quot;"
      providerName="System.Data.SqlClient" />
    <add name="WorkflowDurableInstancingConnection" connectionString="Data Source-
e=<server>;Initial Catalog=<database>;Integrated Secur-
ity=True;Pooling=True;MultipleActiveResultSets=True;Application
Name=&quot;Workflow Composer&quot;"
      providerName="System.Data.SqlClient" />
    <add name="WorkflowTrackingConnection" connectionString="Data Source-
e=<server>;Initial Catalog=&quot;Workflow Tracking&quot;;Integrated Secur-
ity=True;Persist Security
Info=False;Pooling=True;MultipleActiveResultSets=True;Application Name-
e=&quot;Workflow Composer&quot;"
      providerName="System.Data.SqlClient" />
    <add name="CrmDbConnection" connectionString="Data Source=<server>;Initial
Catalog=<database>;Integrated Security=True;Persist Security Info-
o=False;Pooling=True;MultipleActiveResultSets=True;Application Name-
e=&quot;Workflow Composer&quot;"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

When the CRM Web Client URL is populated, an additional appSettings key will be added in the Workflow configuration file.

```
  <appSettings file="">
    <clear />
    <add key="DataMapperAssembly" value="Cmc.CampusLink.BusinessEntities, Culture=neutral, PublicKeyToken=<token>" />
    <add key="ConfigureCampusNexusWcfProxy" value="true" />
    <add key="ConfigureCVueNexusWcfProxy" value="true" />
    <add key="apiKey" value="<key>" />
    <add key="StudentWebClientUrl" value="https://<server>.<domain>:<port>" />
    <add key="CrmWebClientUrl" value="https://<server>.<domain>:<port>/" />
  </appSettings>
```

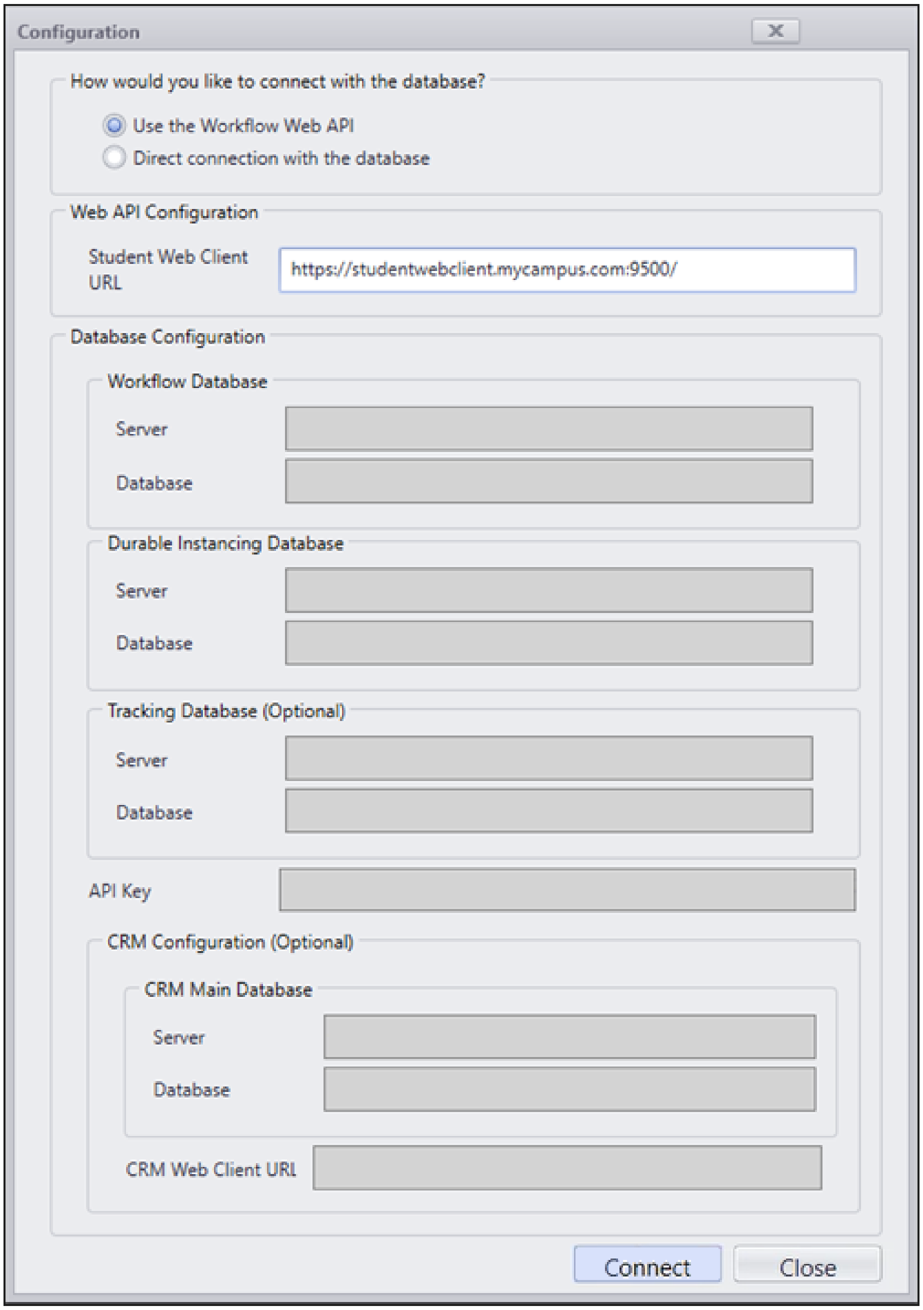
5. Click **Save**.

6. Click **Yes** to proceed. Workflow Composer will restart.

Workflow Web API Connection

If you are using Workflow Composer in an Anthology Cloud 2.0 environment:

1. Select **Use the Workflow Web API**.



Configuration

How would you like to connect with the database?

● Use the Workflow Web API

○ Direct connection with the database

Web API Configuration

Student Web Client URL: https://studentwebclient.mycampus.com:9500/

2. Specify your **Student Web Client URL**, i.e., https://<server>.<domain>:<port>. This URL provides access to the server where the Workflow Web API is deployed.

   The remaining fields are disabled.

   Workflow Composer 3.1 and later supports dual tenancy in Azure AD. This enables Anthology support staff to log in to a customer environment to diagnose an issue. Anthology staff append **account/login/cmc** to the Anthology Student URL value to use a different authentication context for the same environment.

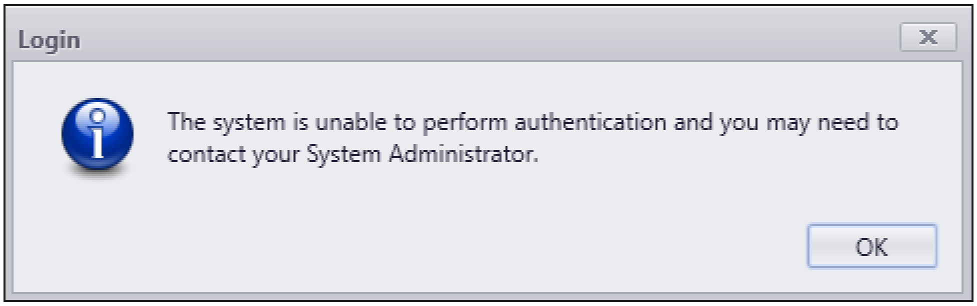
Tenant	Student Web Client URL	Sign in Logo
Azure AD Tenant (Customer)	https://<server>.<domain>:<port>.campusnexus.cloud/	Microsoft
Support Tenant (Anthology Staff)	https://<server>.<domain>:<port>.campusnexus.cloud/account/login/cmc	anthology

3. Click **Save**.

4. Click **Yes** to proceed. Workflow Composer will restart.

When you use the Workflow Web API, you must log in to your Anthology Cloud 2.0 account in the Azure Active Directory (AAD).

In case of a service interruption or incorrect configuration, the following message will be displayed.


Login

The system is unable to perform authentication and you may need to contact your System Administrator.

After you click OK, you can launch Workflow Composer again and the Configuration window will be displayed again. You can change the Web API Configuration and log into your account.

Your user profile in the Anthology Cloud 2.0 AAD must be associated with a role.

- The **Contributor** role allows you to add/publish, delete, and edit workflows.
- The **Reader** role allows you to view workflows.

  As a Reader, you can modify a workflow and save it to the file system. But you cannot publish it. If you try to publish or delete a workflow or persisted instance, Workflow Composer returns the message: *"You are not authorized to perform this action."*

If you are not associated with either role, you will need to contact a System Administrator as you will not have access to the application.

Install Activities and Contracts

After you have configured Workflow Composer, install the Activities and Contracts required for you environment. See Package Manager.

For details about the installation of Workflow Composer, please refer to Installation Manager Help.

Once Workflow Composer is installed, you need to specify whether it accesses the databases via direct connections or via a Workflow Web API.

- In an Anthology Cloud 2.0 environment, configure the Workflow Web API Connection. The Workflow Web API replaces the Citrix connections used previously in cloud environments.
- In on-premise or Azure (non-Anthology Cloud 2.0) environments, configure Direct Database Connections.

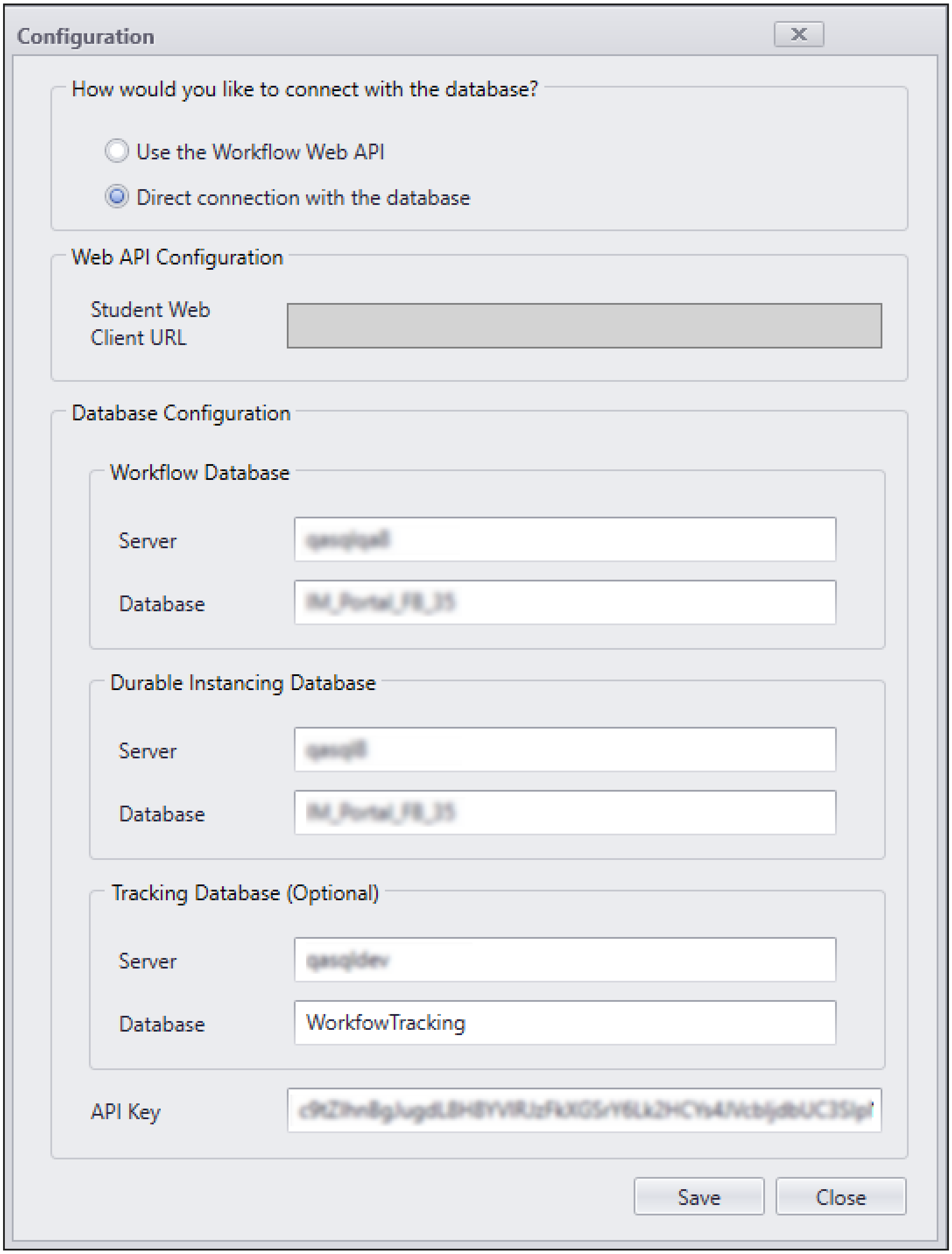
The configuration needs to be done only once when Workflow Composer is installed the first time. The settings are retained during upgrades.

The System tab in the ribbon of Workflow Composer provides a **Configuration** option that enables you to change the initial configuration.

Direct Database Connections

If you are using Workflow Composer with on-premises databases connections:

1. Select **Direct connection with the database**.
2. Specify the server names and database names for your database connections.
   - The **Workflow Database** is the database that supplies values to your workflow activities. It can be an Anthology Student or CampusNexus CRM database.
   - The **Durable Instancing Database** typically uses the same server and database as the Workflow Database.
   - (Optional) The **Tracking Database** is named "WorkflowTracking" by default. It can be on the same server as the Workflow Database and the Durable Instancing Database.
3. In the **API Key** field, specify the key you use to access Anthology workflow Activities and Contracts packages.
4. Click **Save**.



Configuration

How would you like to connect with the database?

○ Use the Workflow Web API

● Direct connection with the database

Web API Configuration

Student Web Client URL

Database Configuration

Workflow Database

Server [redacted]

Database [redacted]

Durable Instancing Database

Server [redacted]

Database [redacted]

Tracking Database (Optional)

Server [redacted]

Database WorkfowTracking

API Key [redacted]

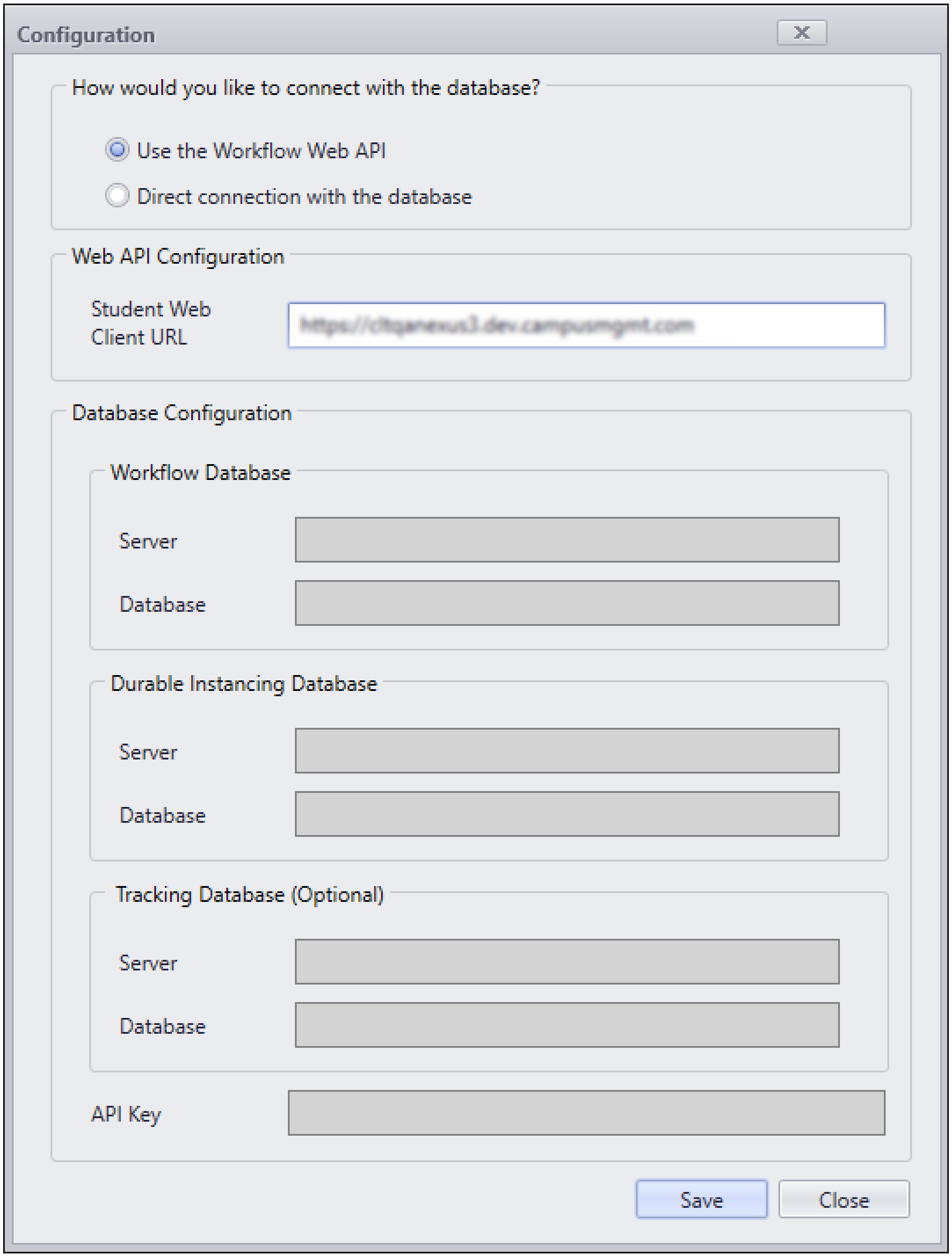
5. Click **Yes** to proceed. Workflow Composer will restart.

Workflow Web API Connection

If you are using Workflow Composer in an Azure cloud environment with Anthology Cloud 2.0:

1. Select **Use the Workflow Web API**.
2. Specify your **Student Web Client URL**, i.e., https://<server>.<domain>:<port>. This URL provides access to the server where the Workflow Web API is deployed.

   The remaining fields are disabled.



Configuration

How would you like to connect with the database?

● Use the Workflow Web API

○ Direct connection with the database

Web API Configuration

Student Web Client URL: [redacted]

Database Configuration

Workflow Database

Server:

Database:

Durable Instancing Database

Server:

Database:

Tracking Database (Optional)

Server:

Database:

API Key:

Workflow Composer 3.1 and later supports dual tenancy in Azure AD. This enables Anthology support staff to log in to a customer environment to diagnose an issue. Anthology staff append **account/login/cmc** to the Anthology Student URL value to use a different authentication context for the same environment.

Tenant	Student Web Client URL	Sign in Logo
Azure AD Tenant (Customer)	https://<server>.<domain>:<port>.campusnexus.cloud/	Microsoft
Support Tenant (Anthology Staff)	https://<server>.<domain>:<port>.campusnexus.cloud/account/login/cmc	CAMPUS MANAGEMENT

3. Click **Save**.

4. Click **Yes** to proceed. Workflow Composer will restart.

When you use the Workflow Web API, you must log in to your Anthology Cloud 2.0 account in the Azure Active Directory (AAD).

In case of a service interruption or incorrect configuration, a message similar to the following will be displayed. You will have the option to return to the Configuration window.

> *“The system is unable to perform authentication. You may need to contact your System Administrator. However, the issue may be the configuration, would you like to review?"*

Your user profile in the Anthology Cloud 2.0 AAD must be associated with a role.

- The **Contributor** role allows you to add/publish, delete, and edit workflows.
- The **Reader** role allows you to view workflows.

  As a Reader, you can modify a workflow and save it to the file system. But you cannot publish it. If you try to publish or delete a workflow or persisted instance, Workflow Composer returns the message: *"You are not authorized to perform this action."*

If you are not associated with either role, you will need to contact a System Administrator as you will not have access to the application.

Install Activities and Contracts

After you have configured Workflow Composer, install the Activities and Contracts required for you environment. See Package Manager.

Contracts

Contracts describe a common data model that can be used to exchange data between service operations from different application domains. The services do not have to share the same architectures or data types. They only need to communicate with each other using the defined data contract.

Event Contracts and Service Contracts enable Anthology to exchange data between applications with different architectures and data models, such as Anthology Student, CampusNexus CRM, and Forms Builder.

- **Event Contracts** define the endpoints that can raise events and respond to events.
- **Service Contracts** specify the operations supported by the service. An operation can be thought of as a Web service method. Each method in the interface corresponds to a specific service operation.

Contracts are available for selection when you create a new workflow in Workflow Designer. The contracts are located in the **Cmc.Nexus.Contracts** library. A contract will exist for each entity/class that exists in the Nexus domain. Some examples of entities are Person, Group, and Organization. Each entity will have a list of events that when raised can invoke a workflow.

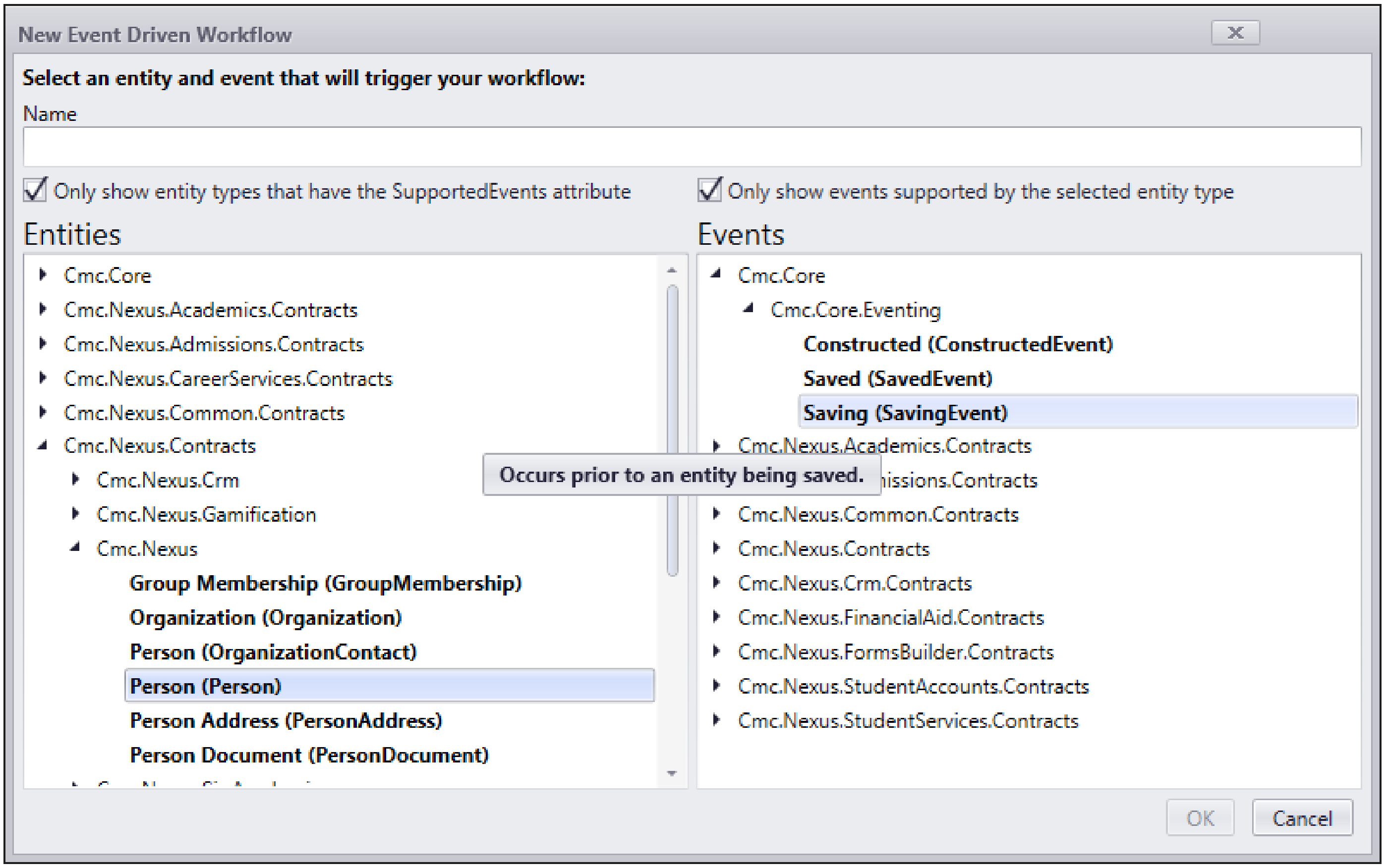
When you create a workflow, you select an **Entity** and an **Event**. The events types available for selection are filtered based on the selected entity.

The entities available for selection are based on the available contracts. Contracts are part of the installed Workflow packages. For more details, see Package Manager.

The option "Only show entity types that have the SupportedEvents attribute" is selected by default. This selection filters events that can trigger workflows. The option "Only show events supported by the selected entity type" is also selected by default. This selection filters events based on the entity type.

Assign a **Name** to the new workflow, click **OK**, and begin building the workflow definition.

Note: Previously, Workflow Composer assigned names using the selected entity and event. Now you can assign any name. The entity and event are displayed when the workflow is published. You can also view the entity and event in the Debug Properties tab next to the Toolbox tab in the Designer.



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

Entities

- Cmc.Core
- Cmc.Nexus.Academics.Contracts
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.CareerServices.Contracts
- Cmc.Nexus.Common.Contracts
- Cmc.Nexus.Contracts
  - Cmc.Nexus.Crm
  - Cmc.Nexus.Gamification
  - Cmc.Nexus
    - **Group Membership (GroupMembership)**
    - **Organization (Organization)**
    - **Person (OrganizationContact)**
    - **Person (Person)**
    - **Person Address (PersonAddress)**
    - **Person Document (PersonDocument)**

Events

- Cmc.Core
  - Cmc.Core.Eventing
    - **Constructed (ConstructedEvent)**
    - **Saved (SavedEvent)**
    - **Saving (SavingEvent)**
- Cmc.Nexus.Academics.Contracts
- issions.Contracts
- Cmc.Nexus.Common.Contracts
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.FormsBuilder.Contracts
- Cmc.Nexus.StudentAccounts.Contracts
- Cmc.Nexus.StudentServices.Contracts

Occurs prior to an entity being saved.

For more information about building workflow definitions, refer to Create Workflows and Sample Workflows.

Create Workflows

Prerequisites

If Workflow Composer is configured to connect directly to the database, **Insert** and **Update** permissions for the following database tables are required:

- WorkflowDefinition
- WorkflowDefinitionVersion

The permissions are required for the logged in user when using integrated security and for the login credentials (username and password) specified if installing via Installation Manager and integrated security is not used.

Also ensure that you have installed the Activities and Contracts packages applicable to your environment. For more information, see Package Manager.

Workflow Types

Workflow Composer can be used to create the following workflow types:

Sequence

- Most common type of workflow.
- Executes a set of child activities according to a single, defined ordering.

Flowchart

- Typically used to implement non-sequential workflows but can be used for sequential workflows if no FlowDecision nodes are used. Flowchart components include:
  - **FlowStep** – models one step of execution in the flowchart (simply a wrapper around a standard activity).
  - **FlowDecision** – branches execution based on a Boolean condition, similar to `If`.
  - **FlowSwitch** – branches execution based on an exclusive switch, similar to `Switch`.

State Machine

- Allows you to model your workflow in an event-driven manner.
- Typically used for human workflow scenarios.
- A state machine can be in one state at any particular time.
  - **Initial State** – represents the starting point of the state machine.
  - **Final State** – represents the completion of the state machine.
  - **Transition** – a directed relationship between two states which represents the response of the state machine to an occurrence of an event.

- **Transition Action** – an activity executed when performing a transition.
- **Entry Action** – an activity executed when entering the state.
- **Exit Action** – an activity executed when exiting the state.
- **Trigger** – a triggering activity that causes a transition to occur.
- **Condition** – a constraint which must evaluate to true after the trigger occurs for the transition to complete.

State machine workflow are used with Forms Builder. See help for Forms Builder 3.x.

Create Workflows with Event Phase

The Cmc.Nexus eventing system was enhanced to raise events for custom service methods in 3 phases (Validation, Execution, and Completion). Workflow Composer 3.0 and later allows you to select the applicable Event Phase for service-based (non-CRUD) events.

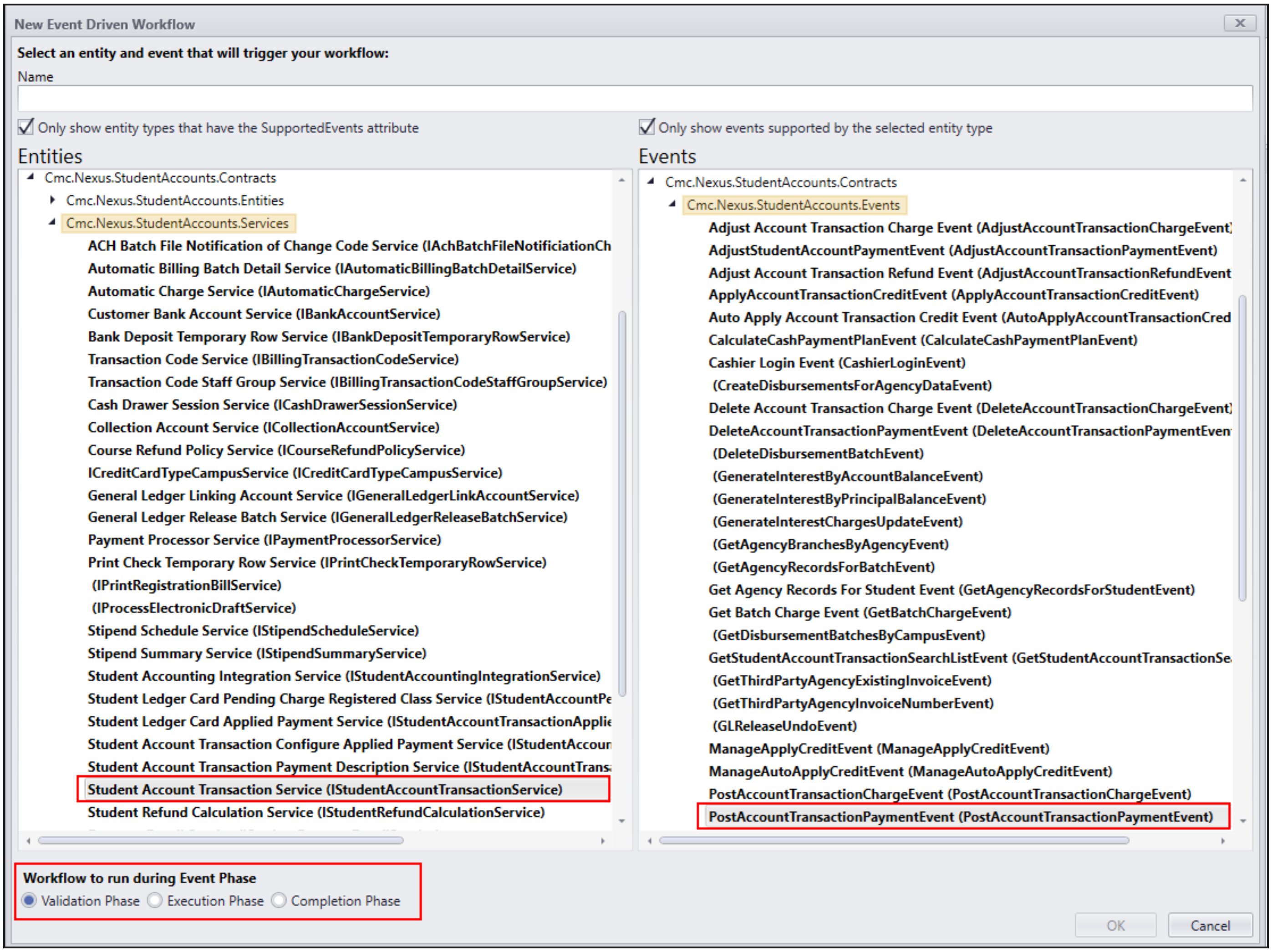
Previously, all workflows were executed during the Execution Phase of a business process. There was no option to add a workflow to be used as validation for an event. For example, it was not possible to inject business logic into a transaction to cancel the execution of a workflow if the custom validation failed. Now, Workflow Composer allows you to select the Validation, Execution, or Completion Phase when creating a workflow.

For any custom service-based workflows created before this enhancement, the workflows will continue to run during the Execution Phase.

Event Phase Selection

Workflows Based on Custom Services

The "New Event Driven Workflow" window in Workflow Composer displays the Event Phase options when you to select a **service-based event** associated with a custom service method, such as the Post Account Transaction Charge Event associated with the Student Account Transaction Service.



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

Entities

- Cmc.Nexus.StudentAccounts.Contracts
  - Cmc.Nexus.StudentAccounts.Entities
  - Cmc.Nexus.StudentAccounts.Services
    - ACH Batch File Notification of Change Code Service (IAchBatchFileNotificiationCh
    - Automatic Billing Batch Detail Service (IAutomaticBillingBatchDetailService)
    - Automatic Charge Service (IAutomaticChargeService)
    - Customer Bank Account Service (IBankAccountService)
    - Bank Deposit Temporary Row Service (IBankDepositTemporaryRowService)
    - Transaction Code Service (IBillingTransactionCodeService)
    - Transaction Code Staff Group Service (IBillingTransactionCodeStaffGroupService)
    - Cash Drawer Session Service (ICashDrawerSessionService)
    - Collection Account Service (ICollectionAccountService)
    - Course Refund Policy Service (ICourseRefundPolicyService)
    - ICreditCardTypeCampusService (ICreditCardTypeCampusService)
    - General Ledger Linking Account Service (IGeneralLedgerLinkAccountService)
    - General Ledger Release Batch Service (IGeneralLedgerReleaseBatchService)
    - Payment Processor Service (IPaymentProcessorService)
    - Print Check Temporary Row Service (IPrintCheckTemporaryRowService)
    - (IPrintRegistrationBillService)
    - (IProcessElectronicDraftService)
    - Stipend Schedule Service (IStipendScheduleService)
    - Stipend Summary Service (IStipendSummaryService)
    - Student Accounting Integration Service (IStudentAccountingIntegrationService)
    - Student Ledger Card Pending Charge Registered Class Service (IStudentAccountPe
    - Student Ledger Card Applied Payment Service (IStudentAccountTransactionApplie
    - Student Account Transaction Configure Applied Payment Service (IStudentAccour
    - Student Account Transaction Payment Description Service (IStudentAccountTrans
    - Student Account Transaction Service (IStudentAccountTransactionService)
    - Student Refund Calculation Service (IStudentRefundCalculationService)

Events

- Cmc.Nexus.StudentAccounts.Contracts
  - Cmc.Nexus.StudentAccounts.Events
    - Adjust Account Transaction Charge Event (AdjustAccountTransactionChargeEvent)
    - AdjustStudentAccountPaymentEvent (AdjustAccountTransactionPaymentEvent)
    - Adjust Account Transaction Refund Event (AdjustAccountTransactionRefundEvent
    - ApplyAccountTransactionCreditEvent (ApplyAccountTransactionCreditEvent)
    - Auto Apply Account Transaction Credit Event (AutoApplyAccountTransactionCred
    - CalculateCashPaymentPlanEvent (CalculateCashPaymentPlanEvent)
    - Cashier Login Event (CashierLoginEvent)
    - (CreateDisbursementsForAgencyDataEvent)
    - Delete Account Transaction Charge Event (DeleteAccountTransactionChargeEvent)
    - DeleteAccountTransactionPaymentEvent (DeleteAccountTransactionPaymentEven
    - (DeleteDisbursementBatchEvent)
    - (GenerateInterestByAccountBalanceEvent)
    - (GenerateInterestByPrincipalBalanceEvent)
    - (GenerateInterestChargesUpdateEvent)
    - (GetAgencyBranchesByAgencyEvent)
    - (GetAgencyRecordsForBatchEvent)
    - Get Agency Records For Student Event (GetAgencyRecordsForStudentEvent)
    - Get Batch Charge Event (GetBatchChargeEvent)
    - (GetDisbursementBatchesByCampusEvent)
    - GetStudentAccountTransactionSearchListEvent (GetStudentAccountTransactionSe
    - (GetThirdPartyAgencyExistingInvoiceEvent)
    - (GetThirdPartyAgencyInvoiceNumberEvent)
    - (GLReleaseUndoEvent)
    - ManageApplyCreditEvent (ManageApplyCreditEvent)
    - ManageAutoApplyCreditEvent (ManageAutoApplyCreditEvent)
    - PostAccountTransactionChargeEvent (PostAccountTransactionChargeEvent)
    - PostAccountTransactionPaymentEvent (PostAccountTransactionPaymentEvent)

Workflow to run during Event Phase

● Validation Phase ○ Execution Phase ○ Completion Phase

Under "Workflow to run during Event Phase" select one of the following:

- Validation Phase
- Execution Phase
- Completion Phase

The selected Event Phase will be embedded into the .xaml file and cannot be modified. Similar to the “Entity” and “Event”, the "Event Phase" cannot be modified once created.

The event pipeline Execution Order is as follows:

A. Execute workflows published to the Validation Phase for the custom event name.
B. If the pipeline is not canceled, execute C# registered handlers for the custom event name.
C. If the pipeline is not canceled, execute workflows published to the Execution Phase for the custom event name.
D. If the pipeline is not canceled, execute workflows published to the Completion Phase for the custom event name.

Workflow event handlers at the Validation Phase are registered at sequence (negative) -1048576 to ensure that they run first. This allows the Validation workflow an opportunity to cancel the process if the Request properties violate any custom business rules.

Workflows event handlers at the Completion Phase are registered at sequence 1048576(1024*1024). Explicitly registering the workflow at this Execution Order ensures that the Completion Phase workflow runs last after all other registered handlers. In the Completion Phase of the event, the args.Response will be populated with the outcome/output of the business process. The process cannot be canceled at this point, but the output could be used to post updates to other entities or integrated systems.

Example Workflow

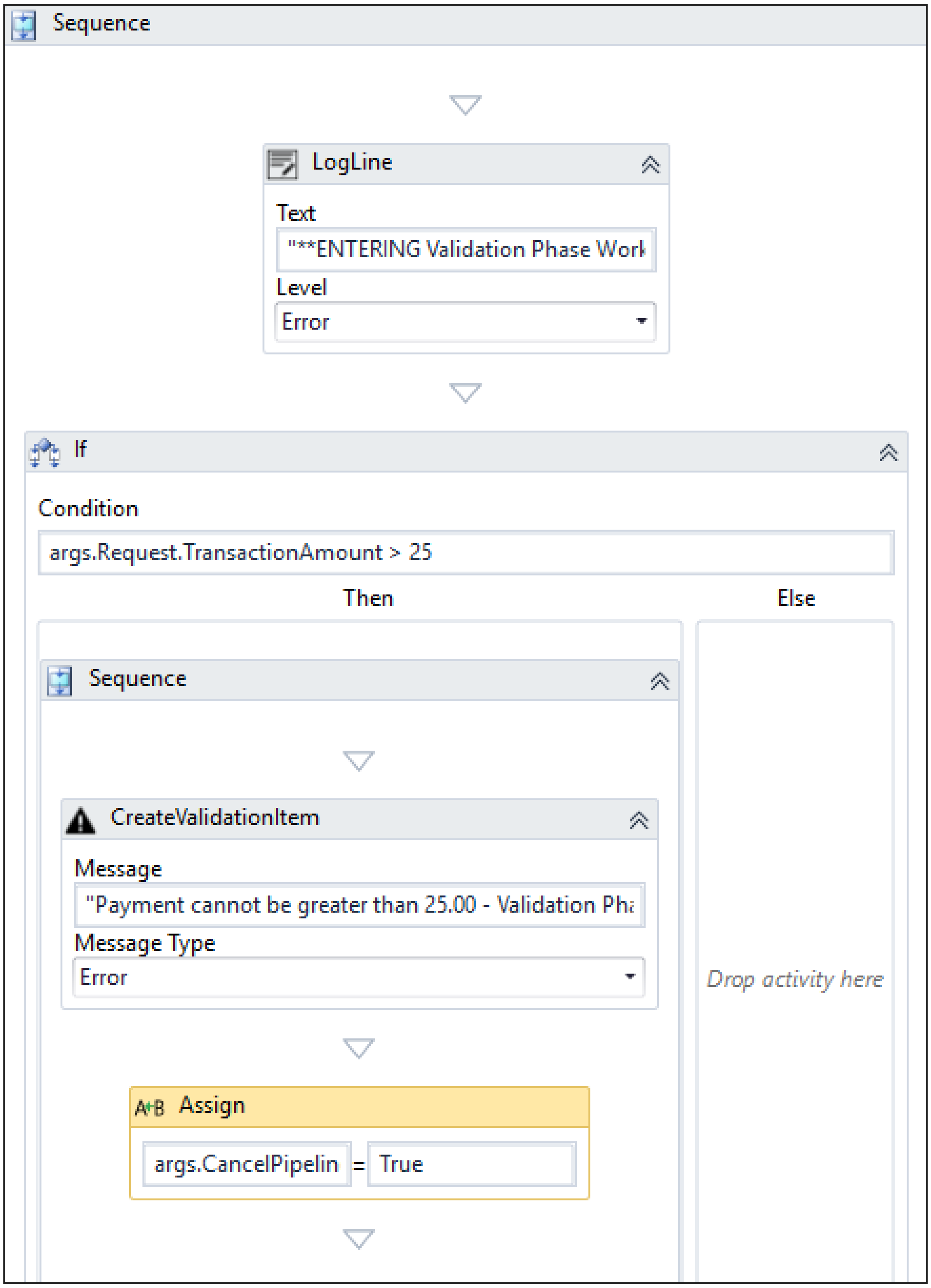
Student Account Transaction Service <> Post Account Transaction Payment Event

We called this service method from a Forms Builder sequence that enables users to make payment online.

Validation Phase

1. When creating a workflow based on this event, select **Validation Phase** for executing the workflow.
2. (Optional) Insert a LogLine activity to mark the beginning of the Validation Phase.
3. Check if the TransactionAmount value is greater than a rule that the institution has for a certain transaction code (e.g., "Books", maximum charge amount is $25.00).

   Use an If activity using Condition = args.Request.TransactionAmount > 25
4. If the TransactionAmount fails the rule, set a Validation Message using a **CreateValidationItem** activity.
5. Insert an Assign activity and specify **args.CancelPipelineExecution = True**.



Text: "**ENTERING Validation Phase Work

Level: Error

Condition: args.Request.TransactionAmount > 25

Message: "Payment cannot be greater than 25.00 - Validation Pha

Message Type: Error

args.CancelPipelin = True

6. Publish the workflow.
7. Since the workflow now runs before anything is posted to the database, if the rule fails and the pipeline is canceled, nothing will be posted to the database, and the Validation Message will be returned.

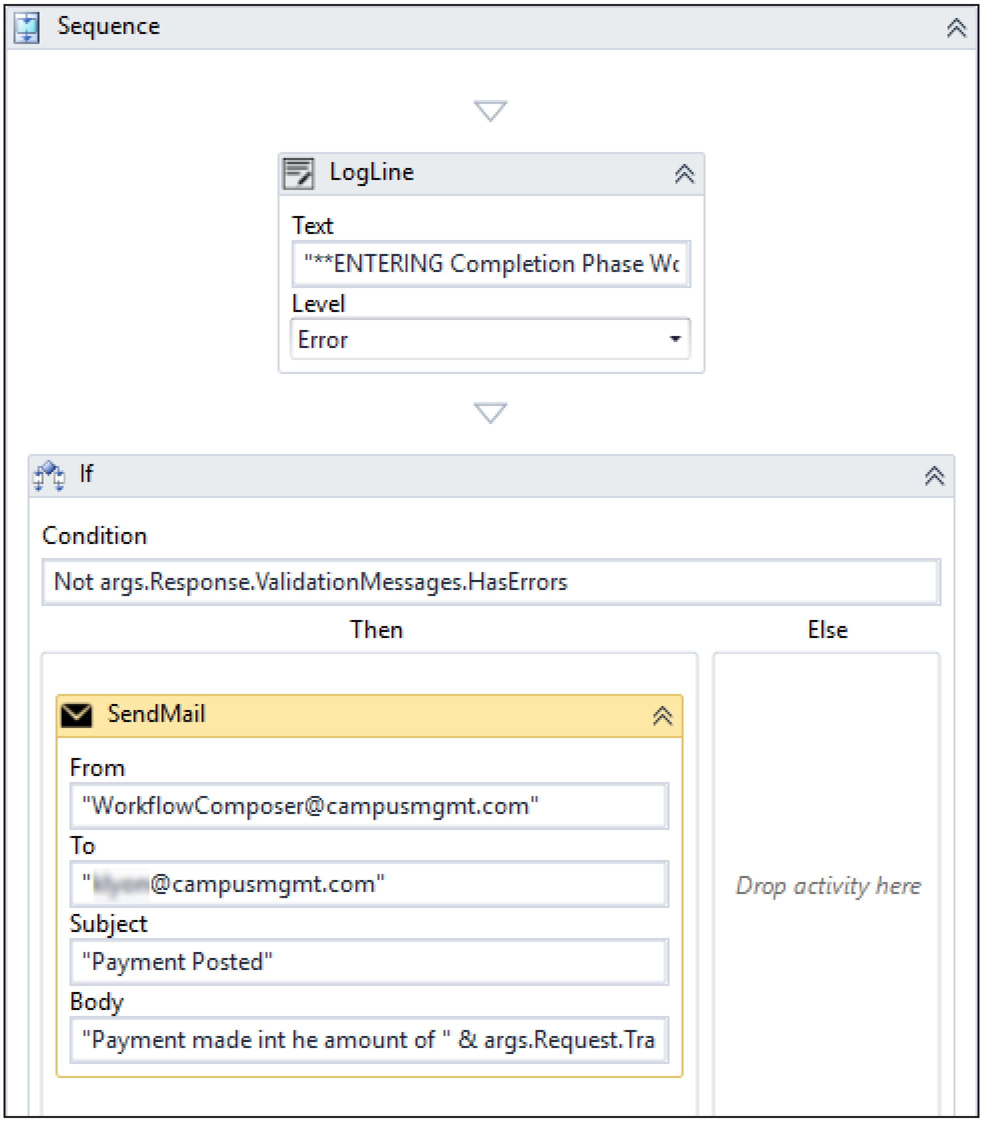
Completion Phase

1. When creating a workflow based on this event, select **Completion Phase** for executing the workflow.
2. (Optional) Insert a LogLine activity to mark the beginning of the Completion Phase.

3. Before adding your Completion Phase activities, make sure the service method was successful.

   This example checks whether the CreateValidationItem activity returned errors using Condition = **Not args.Response.ValidationMessages.HasErrors**

4. If no errors are found, add your Completion Phase activities. This example sends an email message to confirm receipt of the payment.



LogLine

Text: "**ENTERING Completion Phase Wc

Level: Error

Condition: Not args.Response.ValidationMessages.HasErrors

SendMail

From: "WorkflowComposer@campusmgmt.com"

To: "[redacted]@campusmgmt.com"

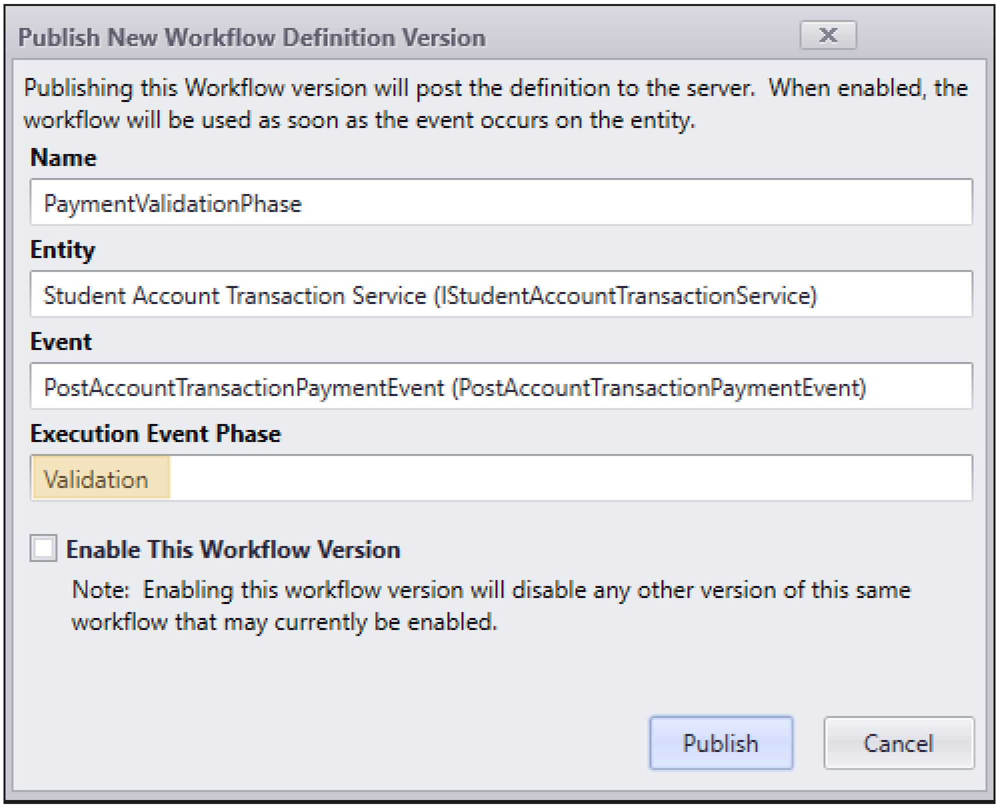
Subject: "Payment Posted"

Body: "Payment made int he amount of " & args.Request.Tra

5. Publish the workflow.

6. The workflow runs after the TransactionAmount passed the max. amount rule and the payment is posted to the database.

When a workflow with Event Phase is published, the selected Event Phase value is visible (but not editable) in the "Publish New Workflow Definition Version" window.



Publish New Workflow Definition Version

Publishing this Workflow version will post the definition to the server. When enabled, the workflow will be used as soon as the event occurs on the entity.

Name

PaymentValidationPhase

Entity

Student Account Transaction Service (IStudentAccountTransactionService)

Event

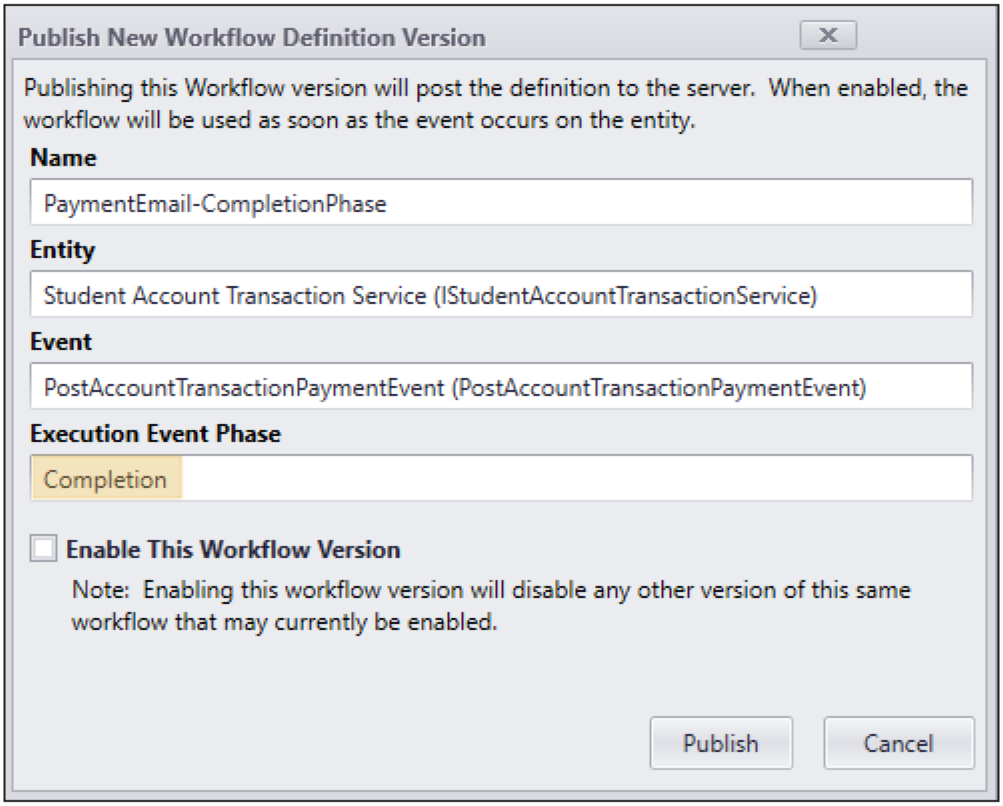
PostAccountTransactionPaymentEvent (PostAccountTransactionPaymentEvent)

Execution Event Phase

Validation

☐ **Enable This Workflow Version**

Note: Enabling this workflow version will disable any other version of this same workflow that may currently be enabled.



Publish New Workflow Definition Version

Publishing this Workflow version will post the definition to the server. When enabled, the workflow will be used as soon as the event occurs on the entity.

Name

PaymentEmail-CompletionPhase

Entity

Student Account Transaction Service (IStudentAccountTransactionService)

Event

PostAccountTransactionPaymentEvent (PostAccountTransactionPaymentEvent)

Execution Event Phase

Completion

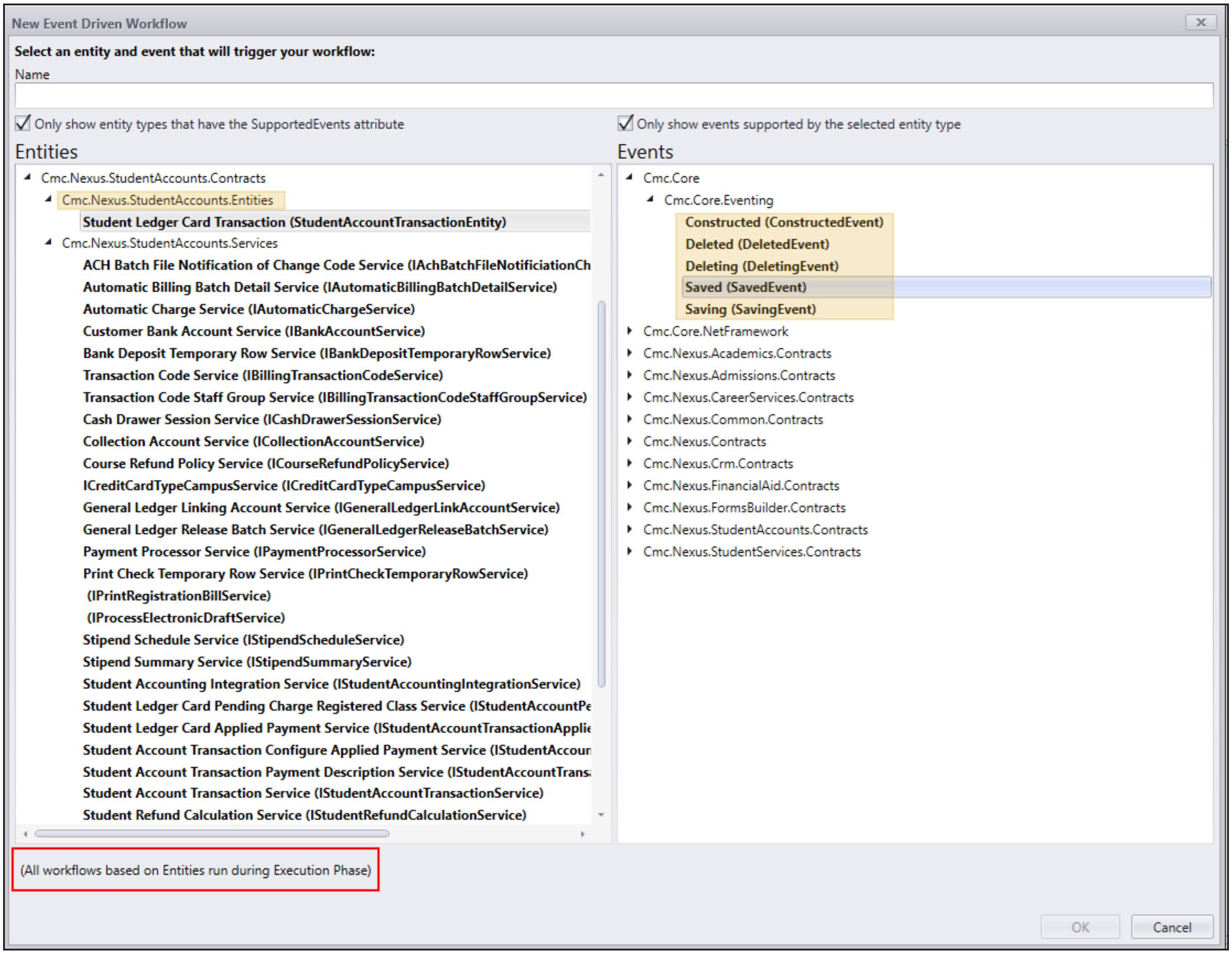
☐ **Enable This Workflow Version**

Note: Enabling this workflow version will disable any other version of this same workflow that may currently be enabled.

Workflows Based on Entities

All events for workflows based on entities will run during the Execution Phase. The default value of "Execution" phase is stored to the workflow and is not editable. The Saving, Saved, Deleting, Deleted, Constructed events continue to execute with Execution Order of 100. This ensures backward compatibility and reduces the complexity of designing workflows for CRUD events. The services for CRUD operations already provide a way to cancel workflow execution using the Saving/Deleting events.

Event phases cannot be selected for entity-based CRUD events, such as Constructed, Deleted, Deleting, Saved, and Saving events.



Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

Entities

- Cmc.Nexus.StudentAccounts.Contracts
  - Cmc.Nexus.StudentAccounts.Entities
    - **Student Ledger Card Transaction (StudentAccountTransactionEntity)**
  - Cmc.Nexus.StudentAccounts.Services
    - **ACH Batch File Notification of Change Code Service (IAchBatchFileNotificiationCh**
    - **Automatic Billing Batch Detail Service (IAutomaticBillingBatchDetailService)**
    - **Automatic Charge Service (IAutomaticChargeService)**
    - **Customer Bank Account Service (IBankAccountService)**
    - **Bank Deposit Temporary Row Service (IBankDepositTemporaryRowService)**
    - **Transaction Code Service (IBillingTransactionCodeService)**
    - **Transaction Code Staff Group Service (IBillingTransactionCodeStaffGroupService)**
    - **Cash Drawer Session Service (ICashDrawerSessionService)**
    - **Collection Account Service (ICollectionAccountService)**
    - **Course Refund Policy Service (ICourseRefundPolicyService)**
    - **ICreditCardTypeCampusService (ICreditCardTypeCampusService)**
    - **General Ledger Linking Account Service (IGeneralLedgerLinkAccountService)**
    - **General Ledger Release Batch Service (IGeneralLedgerReleaseBatchService)**
    - **Payment Processor Service (IPaymentProcessorService)**
    - **Print Check Temporary Row Service (IPrintCheckTemporaryRowService)**
    - **(IPrintRegistrationBillService)**
    - **(IProcessElectronicDraftService)**
    - **Stipend Schedule Service (IStipendScheduleService)**
    - **Stipend Summary Service (IStipendSummaryService)**
    - **Student Accounting Integration Service (IStudentAccountingIntegrationService)**
    - **Student Ledger Card Pending Charge Registered Class Service (IStudentAccountPe**
    - **Student Ledger Card Applied Payment Service (IStudentAccountTransactionApplie**
    - **Student Account Transaction Configure Applied Payment Service (IStudentAccour**
    - **Student Account Transaction Payment Description Service (IStudentAccountTrans**
    - **Student Account Transaction Service (IStudentAccountTransactionService)**
    - **Student Refund Calculation Service (IStudentRefundCalculationService)**

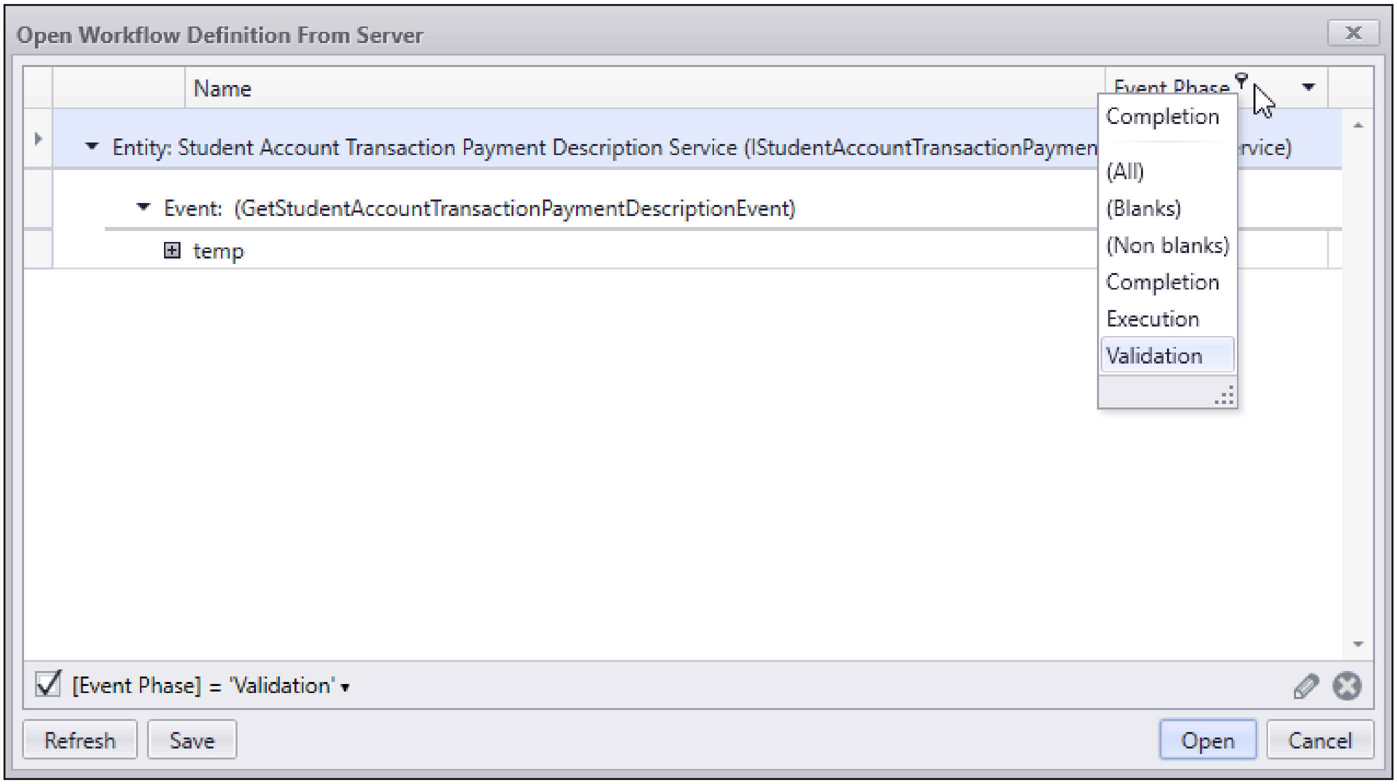
Events

- Cmc.Core
  - Cmc.Core.Eventing
    - **Constructed (ConstructedEvent)**
    - **Deleted (DeletedEvent)**
    - **Deleting (DeletingEvent)**
    - **Saved (SavedEvent)**
    - **Saving (SavingEvent)**
- Cmc.Core.NetFramework
- Cmc.Nexus.Academics.Contracts
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.CareerServices.Contracts
- Cmc.Nexus.Common.Contracts
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.FormsBuilder.Contracts
- Cmc.Nexus.StudentAccounts.Contracts
- Cmc.Nexus.StudentServices.Contracts

(All workflows based on Entities run during Execution Phase)

Event Phase Filter

When opening a workflow from the server, you can **filter** workflows by Event Phase.



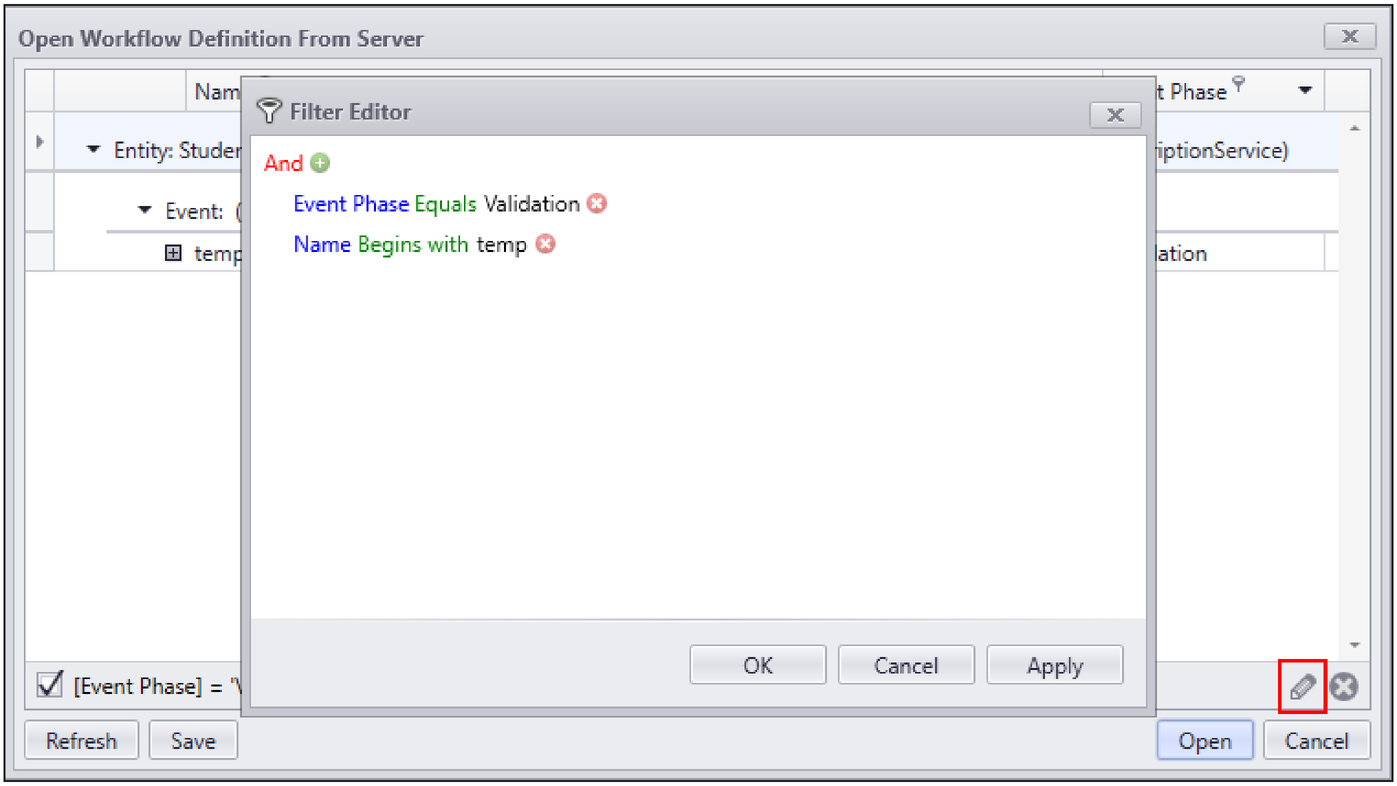
Entity: Student Account Transaction Payment Description Service (IStudentAccountTransactionPaymen...rvice)

Event: (GetStudentAccountTransactionPaymentDescriptionEvent)

temp

☑ [Event Phase] = 'Validation'

You can **edit the filter** to narrow the search results.



And

Event Phase Equals Validation

Name Begins with temp

☑ [Event Phase] = 'V

Exception Handling

Exception handling refers to exceptions that are thrown from workflows as well as responses from the Anthology API services when the Windows Communication Foundation (WCF) service returns validation messages.

Workflow Design Requires Exception Handling

The user who creates workflows is responsible for catching exceptions. Any unexpected and uncaught exceptions will abort workflows. For the guidelines on exception handling within workflows, refer to Coding for Activity Errors.

Exception Message Queues

Workflow exception messages are queued. Queues ensure that reliable messaging can occur between a client and a service, even if the service is not available at the time of communication.

Anthology uses dead-letter queue and poison message handling provided by the WCF framework. For more information, see http://msdn.microsoft.com/en-us/library/ms789035(v=vs.110).aspx.

If an error is found in a workflow, the message queue flags exceptions as `EXTERNAL_DeadLetterQueue`.

A log file on the application server provides detailed information about `Compiler errors` in the workflow.

The failed messages in the dead-letter queue are tried again. If the exception cannot be resolved, the first entry is cleared from the dead letter queue. This ensures that the service broker is not locked in case of an exception. Users can retry the transaction after the error is cleared.

In addition, the Service Broker queue processor code implements a `Trace.WriteLine` mechanism to handle failed messages:

```
/// <summary>
///     Handle failed messages
/// </summary>
/// <param name="message"></param>
/// <param name="messageType"></param>
/// <param name="con"></param>
/// <param name="errorInfo"></param>
     public static void SaveFailedMessage(string message, string messageType, SqlConnection con,
Exception errorInfo)
     {
        Trace.WriteLine("CVueExternalMessageProcessor Received Failed Message");
     }
```

The default behavior of Service Broker is to disable a queue after the same message has thrown an exception five times. Anthology provides a setting in the `config` file that prevents shutting down the queue.

```
     <setting name="ShutdownQueueOnError" serializeAs="String">
     <value>False</value>
     </setting>
```

When this setting is set to True, the queue gets disabled. When this setting is set to False, the queue does not get disabled. False is the recommended setting.

Helpful Hints

The following hints may help when you begin creating and testing workflows.

Use Conditions

Workflows should start with a condition that determines if or when a workflow is executed. It is important to use conditions because all workflows that are stored on a workstation are active. Proper conditions prevent conflicting or unintended changes to the database.

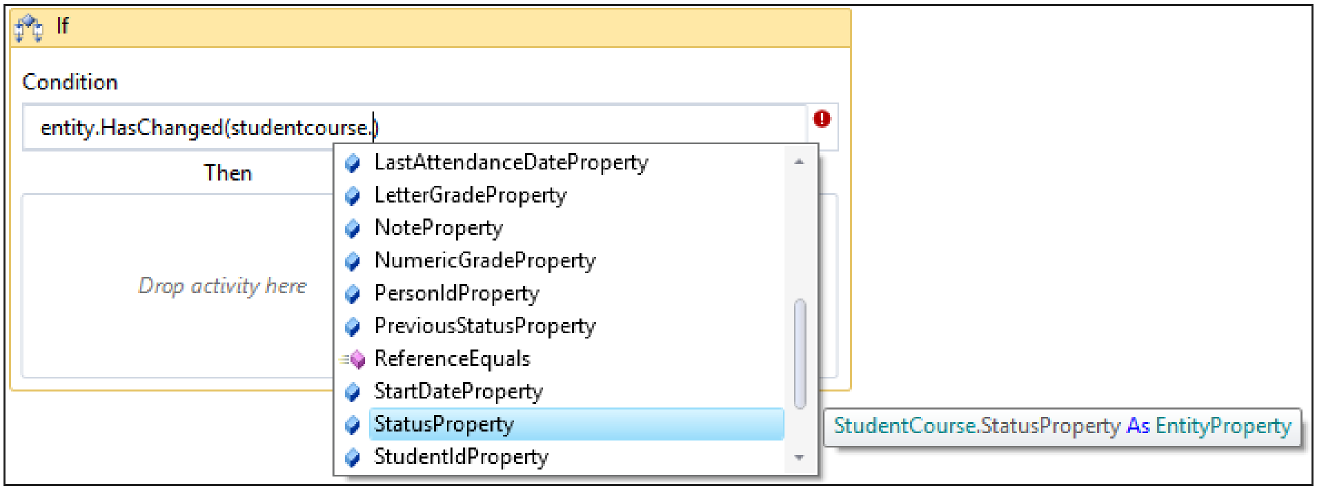
Check for Record Inserts and Changes

When working with workflows, it is important to understand that many of the records that are checked in the workflow will have numerous updates from different sources for different reasons and the workflow will be triggered multiple times. To ensure that the workflow is executed only when a specific value is changed, you can use conditions to check the `EntityState` property or the `HasChanged` method on the entity.

Examples

- `entity.HasChanged("Veteran")` – checks if the veteran flag on a Person record was modified.
- `entity.Prospects(0).HasChanged(“LeadTypeId”)` – checks if the identifier of a Person record was modified indicating that a new record was inserted.
- `entity.HasChanged(StudentCourse.StatusProperty)` – checks if the Status property on the Student Course entity has changed.

  In a condition statement for any entity you can select all the available properties that you are looking for to have changed. In this example the entity is StudentCourse and the StatusProperty is selected.

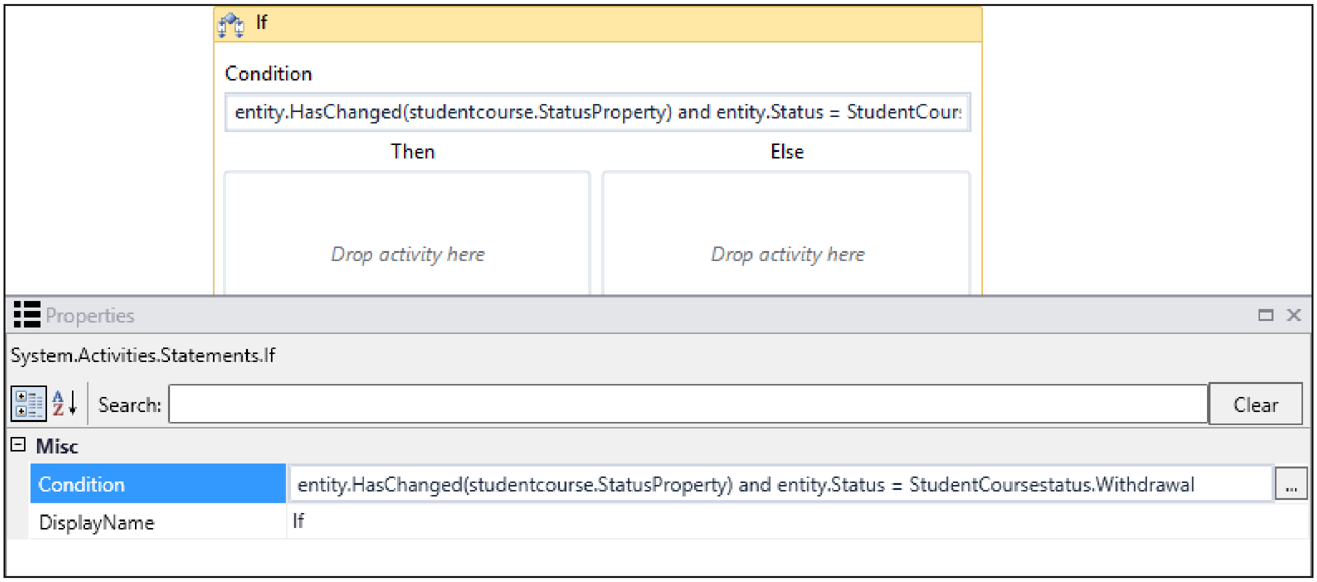
  

  entity.HasChanged(studentcourse.)

  StudentCourse.StatusProperty As EntityProperty

  To determine if a Student Course Status changed to "Withdrawal" (= "Drop" in Anthology Student, specify the following condition:

```
entity.HasChanged(studentcourse.StatusProperty) and entity.Status = Stu-
dentCourseStatus.Withdrawal
```



Condition: entity.HasChanged(studentcourse.StatusProperty) and entity.Status = StudentCour

System.Activities.Statements.If

Condition	entity.HasChanged(studentcourse.StatusProperty) and entity.Status = StudentCoursestatus.Withdrawal
DisplayName	If

As a general rule do not use Save type activities in Saving events, only Saved events.

You can also use the `entity.HasChanged` condition to prevent infinite loops in the workflow.

The `EntityState` property applies to the entity to which it belongs. For example, the `Person` entity did not change, but one of its child entities (`Prospects`) did. If you check the `entity.Prospects(0).EntityState`, it should indicate `Modified`.

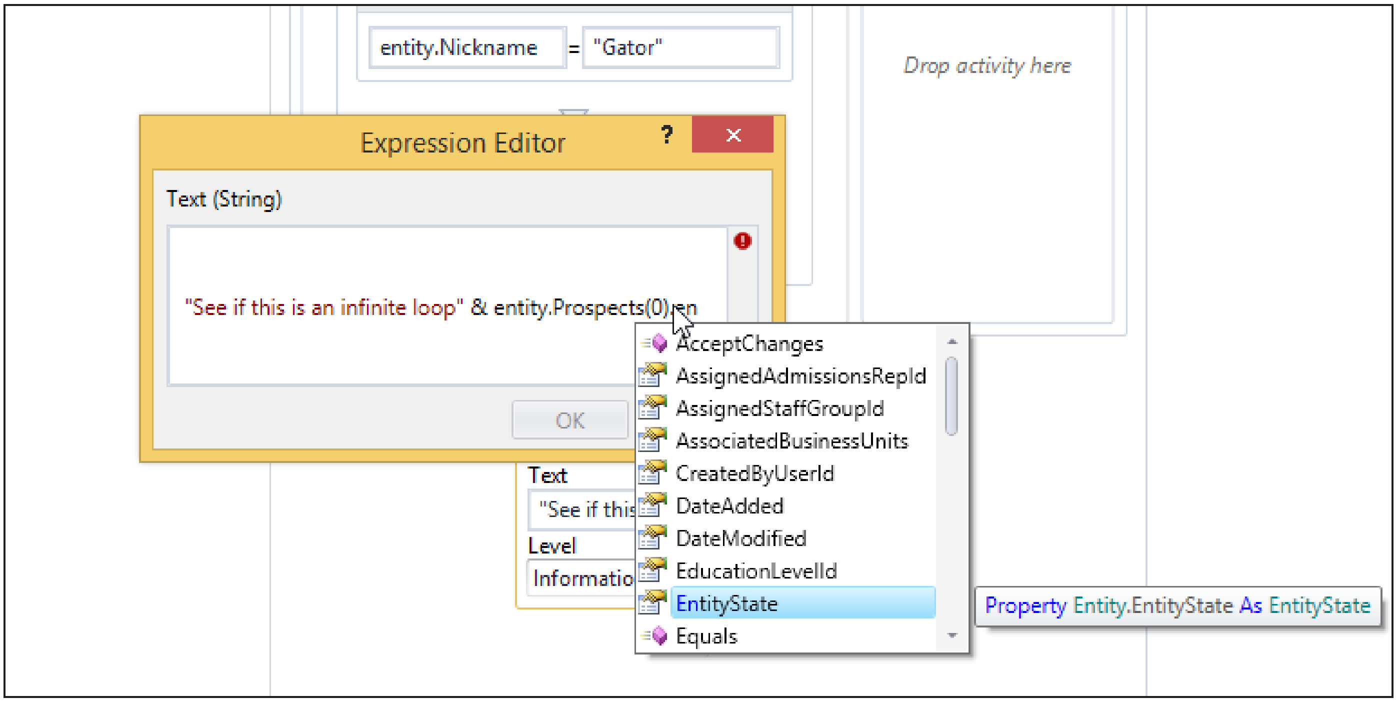
The `EntityState` property and the `HasChanged()` method are intended for different uses and have specific meanings. The following are examples for a `Person` entity:

- `entity.HasChanged()` – indicates if any direct properties of the `Person` entity have changed. This does not check any child entities or collections.
- `entity.HasChanged(true)` – checks the `Person` entity plus any child entities and collections. If any property on the `Person` entity, or any of the entities in the collections (`Students, Prospects`) have changed, it will return `true`. Use `entity.HasChanged(true)` in workflows to determine if anything has changed within the model.
- `entity.Prospects(0).HasChanged()` – returns `true` if the first `Prospects` child entity of the `Person` has any changes.
- `entity.Prospects(0).EntityState` – returns one of three values `Added`, `Modified`, or `Removed` and only applies to the first `Prospects` entity in the `Prospects` collection.

For an activity that adds a record to an entity, every property will be dirty because the values are set from null to something else or to an empty string. Therefore, you should check the `EntityState` in your workflow to determine if a record is added. Insert a condition similar to the following:

```
If [ entity.EntityState = Cmc.Core.EntityModel.EntityState.Added ]
```

- `entity.EntityState` – is an enumeration and contains one of three values `Added`, `Modified`, or `Removed`. This gives the workflow developer more information about what has happened to the entity during the process. This is specific to the entity to which the `EntityState` belongs.



entity.Nickname = "Gator"

Text (String)

```
"See if this is an infinite loop" & entity.Prospects(0).en
```

Text: "See if this

Level: Informatio

Prevent Loops

Be careful not to create loops in your workflow statements.

Examples:

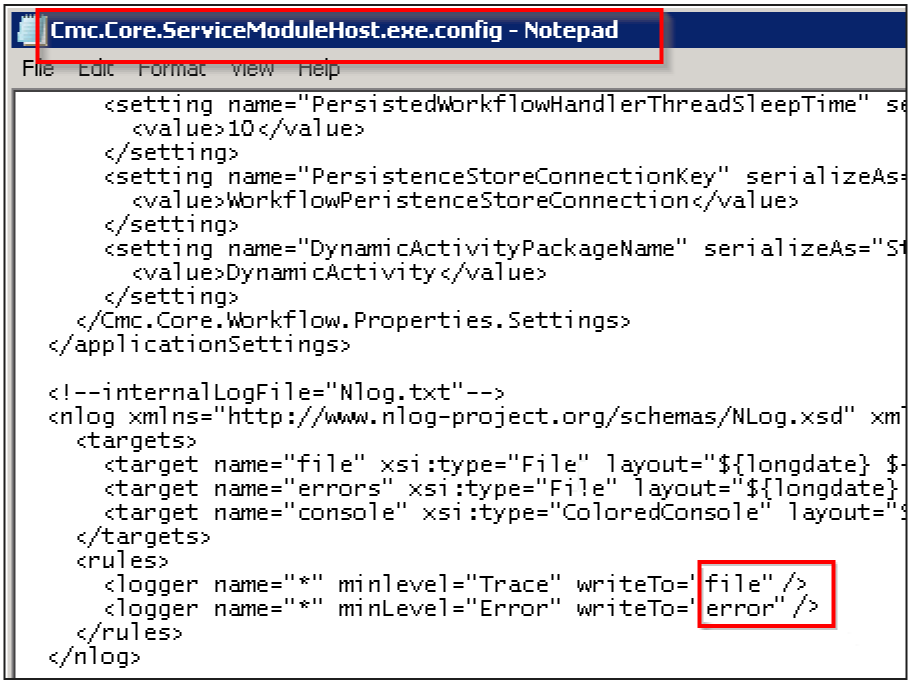
- If a workflow is triggered by a saving event, don't use a Save activity within the workflow.
- If a workflow is triggered by the posting of a charge, don't use a CreateCharge activity within the workflow.

Test Workflows for Saved Events

Although Workflow is distributed with logging turned off, you might want to enable logging during the workflow design phase. See NLog for details about the logging configuration.

It is a good practice to insert at least one LogLine activity in workflows for Saved events. The LogLine text will appear in the event log immediately after the event is raised.

Note: The LogLine activity requires the Cmc.Core.ServiceModuleHost.exe.config file to be set up to log to file and error as shown below.


Cmc.Core.ServiceModuleHost.exe.config - Notepad
File Edit Format View Help

```
    <setting name="PersistedWorkflowHandlerThreadSleepTime" se
      <value>10</value>
    </setting>
    <setting name="PersistenceStoreConnectionKey" serializeAs=
      <value>WorkflowPeristenceStoreConnection</value>
    </setting>
    <setting name="DynamicActivityPackageName" serializeAs="St
      <value>DynamicActivity</value>
    </setting>
  </Cmc.Core.Workflow.Properties.Settings>
</applicationSettings>

<!--internalLogFile="Nlog.txt"-->
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd" xm
  <targets>
    <target name="file" xsi:type="File" layout="${longdate} $
    <target name="errors" xsi:type="File" layout="${longdate}
    <target name="console" xsi:type="ColoredConsole" layout="$
  </targets>
  <rules>
    <logger name="*" minlevel="Trace" writeTo="file" />
    <logger name="*" minLevel="Error" writeTo="error" />
  </rules>
</nlog>
```

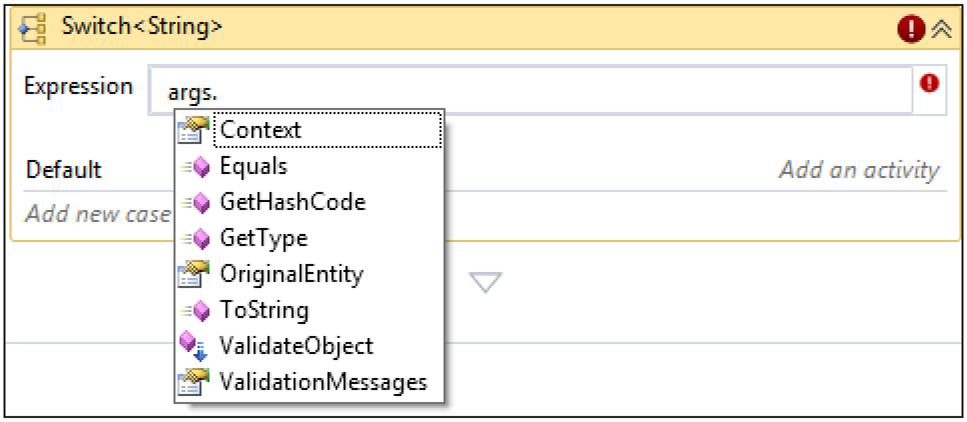
Check the `date.errors.log` file regularly for any errors in your workflows. For more information, see [Event Logs](#).

Alternatively, you can test workflows for Saved events by including a Contact Manager CreateTask activity. You can confirm that the workflow was executed by checking the Contact Manager UI.

Filter Events Based on Event Source

Every event has arguments. The arguments can be viewed in Intellisense by typing **args.** in the Workflow Designer.

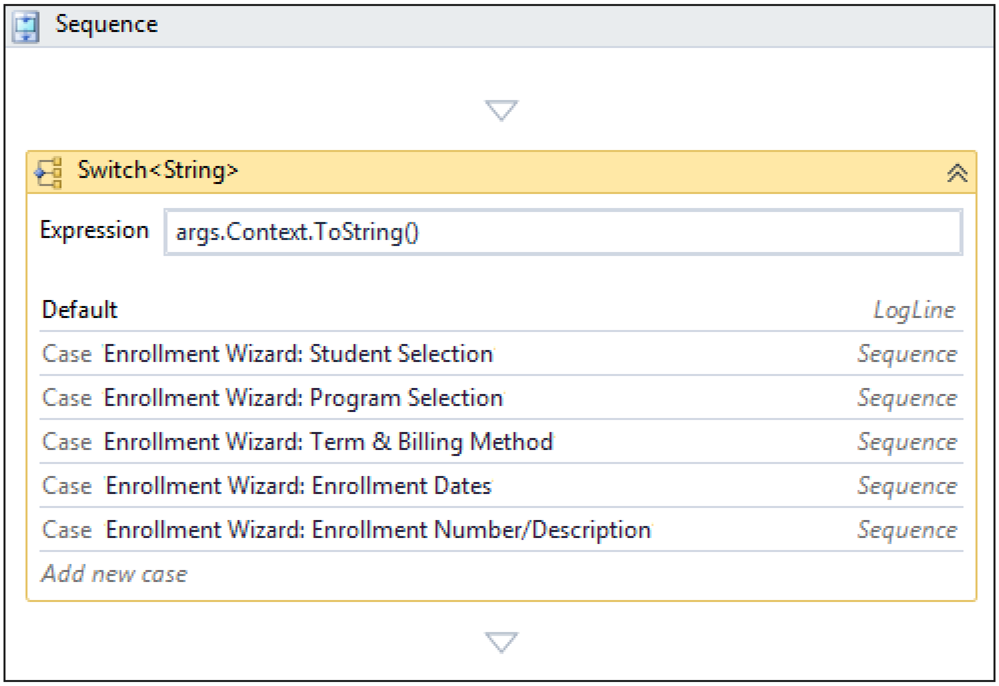
Event arguments have a connection context that specifies where the transaction came from. The context information can be used to filter events. For example, you can set up a filter to handle only events that came from a specific database trigger.



Context Property

The Context property is a string that is set in the code when an event is raised. You can access the Context property in the Workflow Designer, for example, when you specify `arg.` in the Expression field of a Switch activity.

The Context property is useful when a workflow is associated with a sequence of forms such as the Enrollment Wizard in Anthology Student. When the user clicks Next after completing Step 1 in the Enrollment Wizard, a Person Saving event is raised and the Context is set to a string, in this case, “Enrollment Wizard: Student Selection”. You can use a conditional statement to check the value of Context and validate fields in Step 1. Within the workflow, as you proceed through validating fields in the sequence of steps, check the Context string using each Case of the Switch activity. See the sample workflow Enrolling Students Using the Enrollment Wizard.



Sequence

Switch<String>

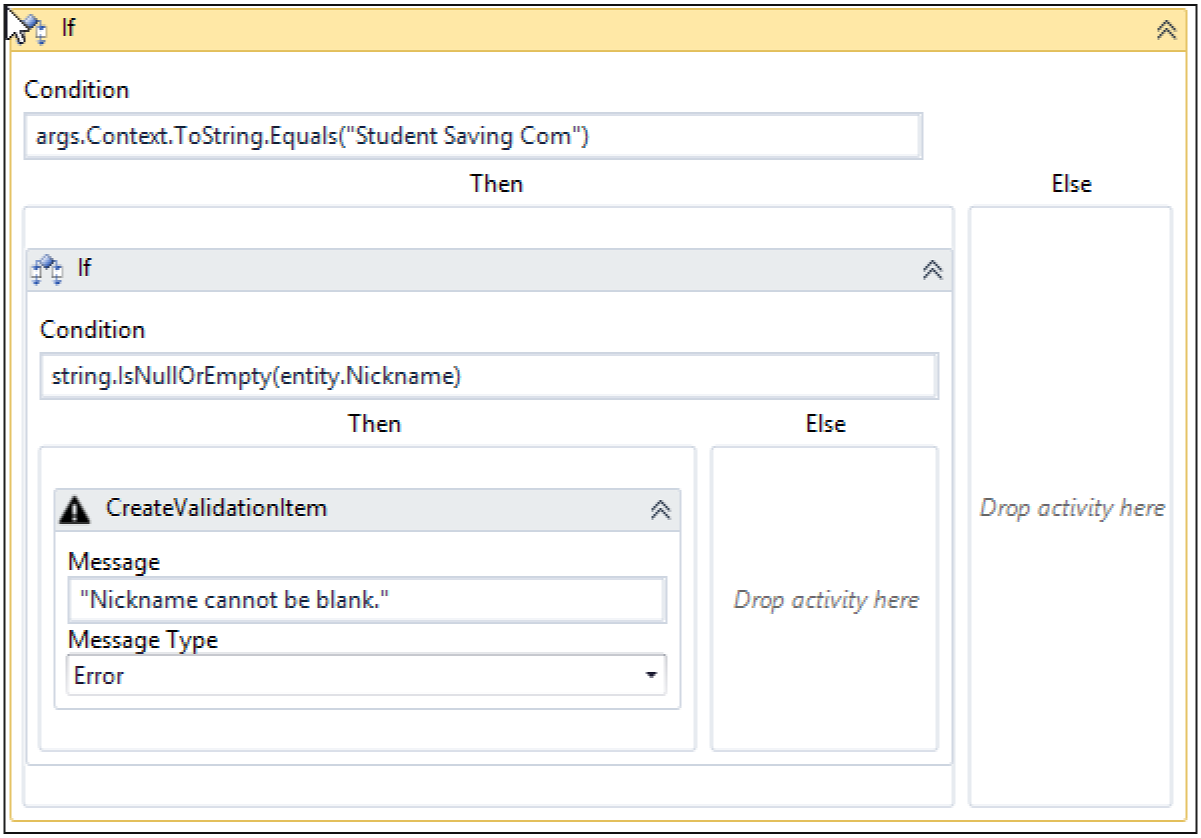
Expression: args.Context.ToString()

Default		LogLine
Case	Enrollment Wizard: Student Selection	Sequence
Case	Enrollment Wizard: Program Selection	Sequence
Case	Enrollment Wizard: Term & Billing Method	Sequence
Case	Enrollment Wizard: Enrollment Dates	Sequence
Case	Enrollment Wizard: Enrollment Number/Description	Sequence

Add new case

Without the Context property, if the workflow validated a property that was picked in Step 4 of the wizard and the event was triggered for Step 1, unexpected behavior or null reference exceptions may occur.

Note that the Enrollment Wizard uses a Person Saving entity contract, so if you have a validation for the Student Master form (e.g., on Nickname) you should also add a context sensitive `If` statement in that workflow. Context in that case is “Student Saving Com”. Otherwise some validation you have for the Student Master could show up on every step of the Enrollment Wizard on fields that are not even available there.



If

Condition

args.Context.ToString.Equals("Student Saving Com")

Then

If

Condition

string.IsNullOrEmpty(entity.Nickname)

Then

CreateValidationItem

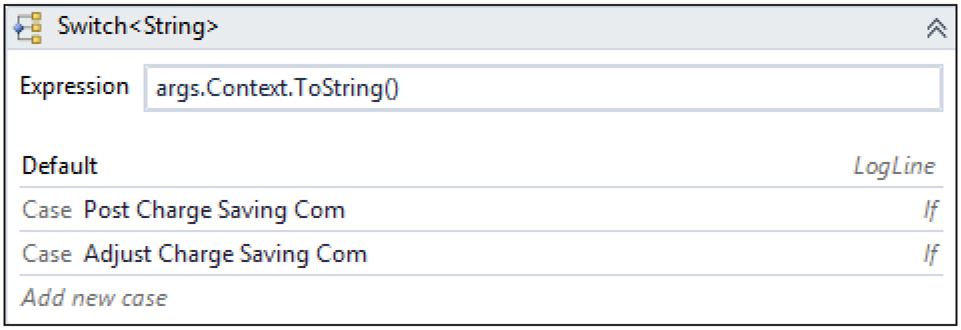
Message

"Nickname cannot be blank."

Message Type

Error

Another use case for the Context property are workflows that deal with PostCharge or AdjustCharge transaction. The Context property can be used to determine the type of event.



Switch<String>

Expression args.Context.ToString()

Default	LogLine
Case Post Charge Saving Com	If
Case Adjust Charge Saving Com	If

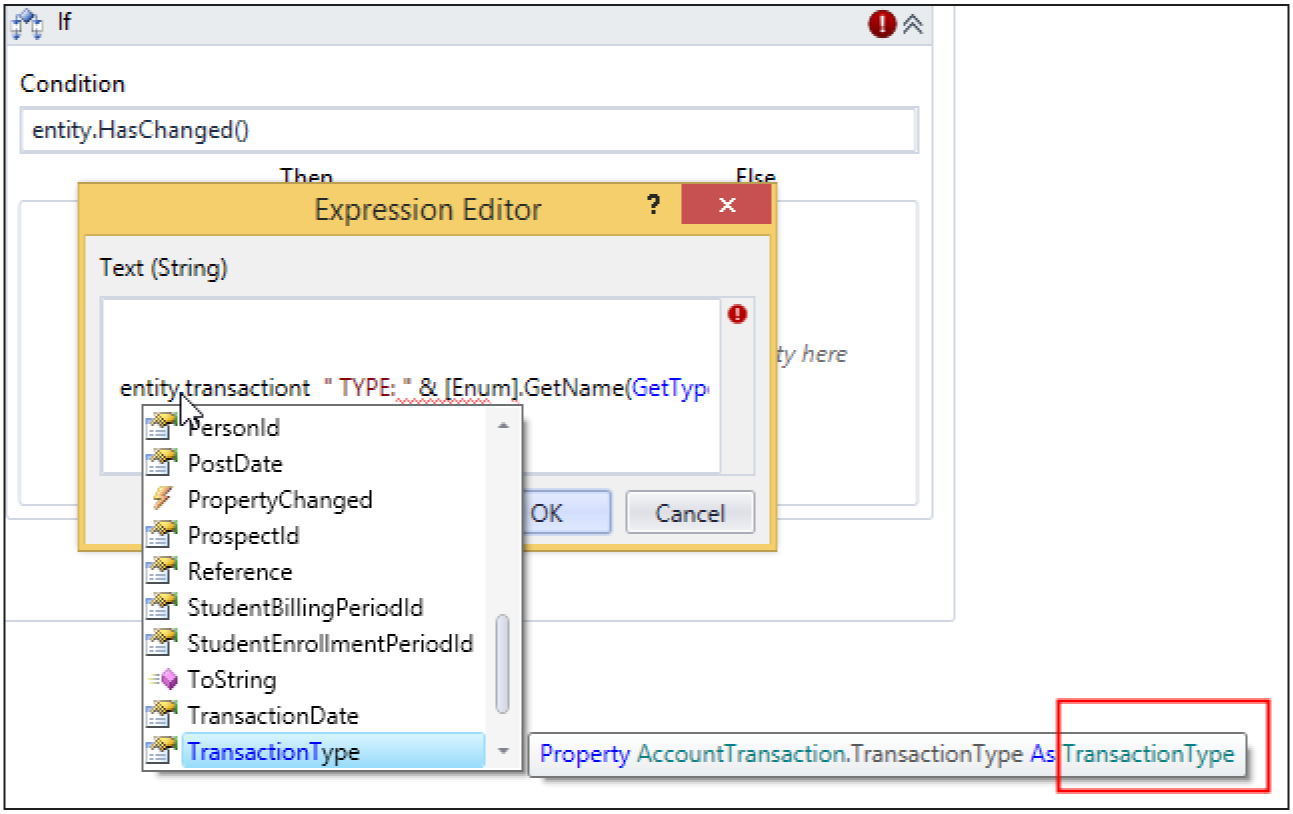
Retrieve an Enum Value

For entities containing enumerations (i.e., a predefined list of values), use the Enum.GetName method to retrieve an enum value.

Example:

The following expression retrieves the value of the TransactionType enumeration in the Cmc.Nexus.Sis.StudentAccounts contract:

```
[Enum].GetName(GetType(Cmc.Nex-
us.Sis.StudentAccounts.TransactionType),entity.TransactionType)
```



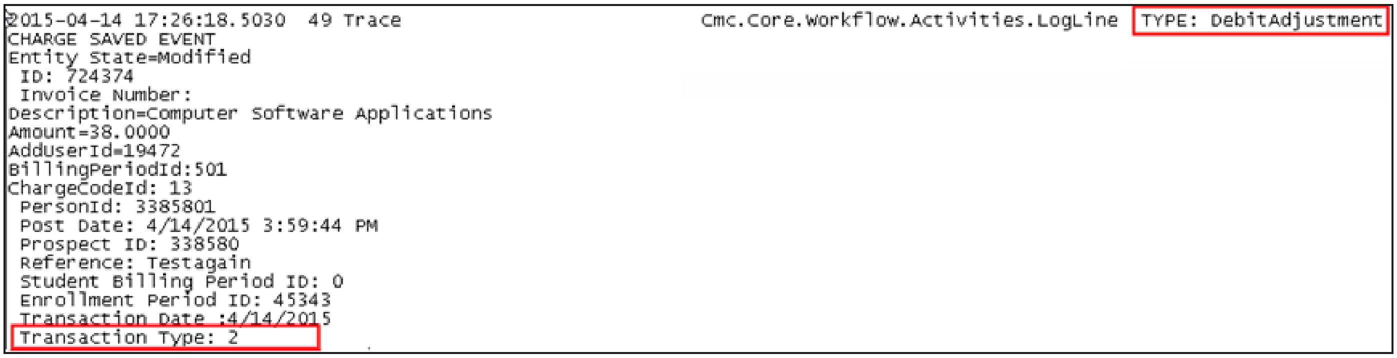
```
entity.HasChanged()
```

```
entity.transactiont  " TYPE: " & [Enum].GetName(GetTyp
```

```
Property AccountTransaction.TransactionType As TransactionType
```

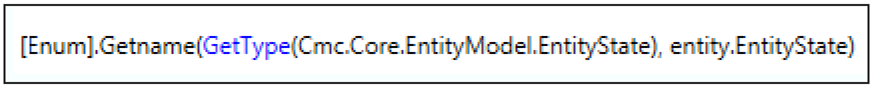
In the case of the `TransactionType` enumeration, the `Enum.GetName` method enables you to capture the Transaction Type value and perform another workflow activity when this value is found.

The log shows the mapping of the TransactionType enum value of "2" to the Transaction Type of "DebitAdjustment".



```
2015-04-14 17:26:18.5030  49 Trace                         Cmc.Core.Workflow.Activities.LogLine  TYPE: DebitAdjustment
CHARGE SAVED EVENT
Entity State=Modified
 ID: 724374
 Invoice Number:
Description=Computer Software Applications
Amount=38.0000
AddUserId=19472
BillingPeriodId:501
ChargeCodeId: 13
 PersonId: 3385801
 Post Date: 4/14/2015 3:59:44 PM
 Prospect ID: 338580
 Reference: Testagain
 Student Billing Period ID: 0
 Enrollment Period ID: 45343
 Transaction Date :4/14/2015
 Transaction Type: 2
```

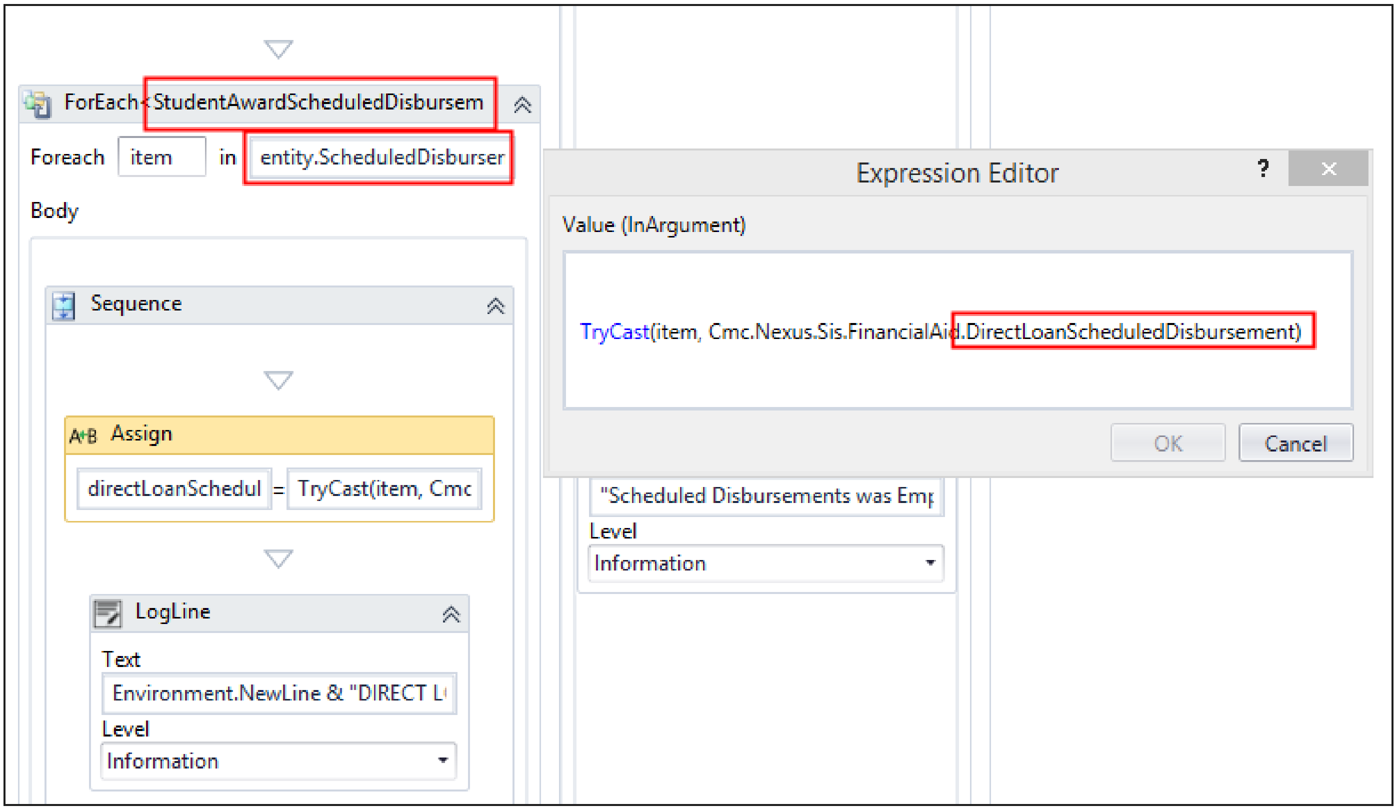
Another commonly used property to retrieve an enumeration is `EntityState` as shown below:



```
[Enum].Getname(GetType(Cmc.Core.EntityModel.EntityState), entity.EntityState)
```

Type Casting

You can convert data types using the TryCast operator. The example below shows how the Loan ScheduledDisbursement data type can be converted to the more specific DirectLoanScheduledDisbursement.

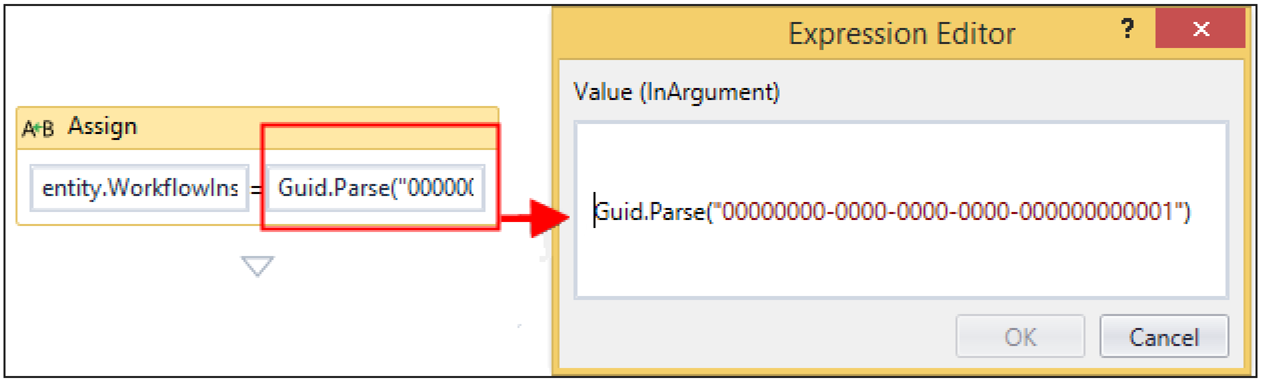


ForEach<StudentAwardScheduledDisbursem
Foreach item in entity.ScheduledDisburser
Body
Sequence
Assign
directLoanSchedul = TryCast(item, Cmc
LogLine
Text
Environment.NewLine & "DIRECT L
Level
Information
"Scheduled Disbursements was Emp
Level
Information
Expression Editor
?
Value (InArgument)
OK
Cancel

```
TryCast(item, Cmc.Nexus.Sis.FinancialAid.DirectLoanScheduledDisbursement)
```

Clear a Workflow Instance Id

To clear a Workflow Instance Id value in a workflow, use the following syntax:

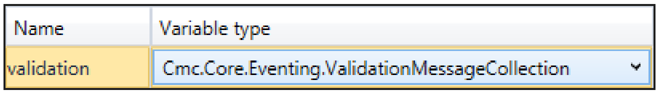


```
Guid.Parse("00000000-0000-0000-0000-000000000001")
```

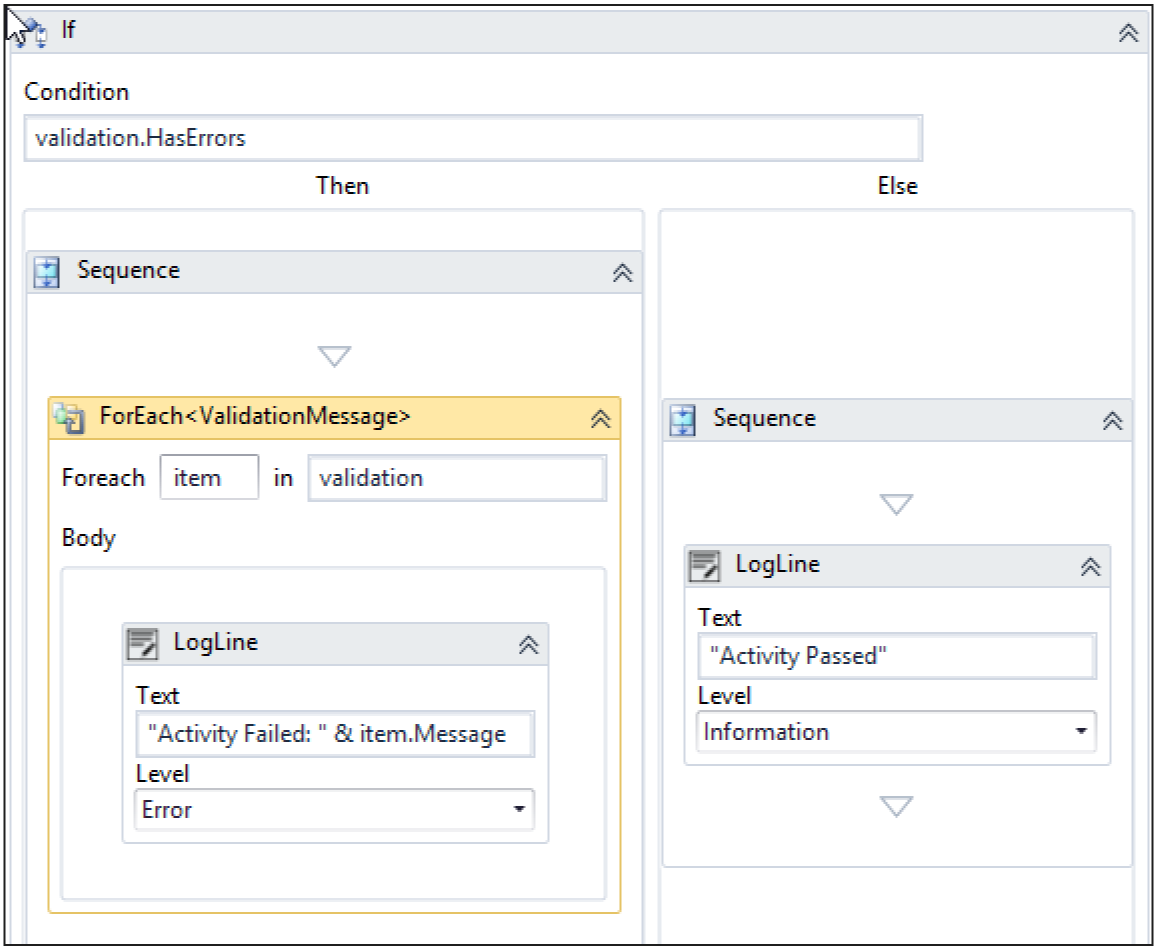
Note: The API does not allow you to set the Guid value to all 0s. Therefore, the 1 appears at the end.

Capture Validation Errors

In activities that provide a ValidationMessages field defined as `InOutArgument<ValidationMessageCollection>`, you can create a variable of type *ValidationMessageCollection* and use the variable to capture error messages as shown in the example below, where the name of the variable is "validation".



Name	Variable type
validation	Cmc.Core.Eventing.ValidationMessageCollection



Condition: validation.HasErrors

Foreach item in validation

Text: "Activity Failed: " & item.Message
Level: Error

Text: "Activity Passed"
Level: Information

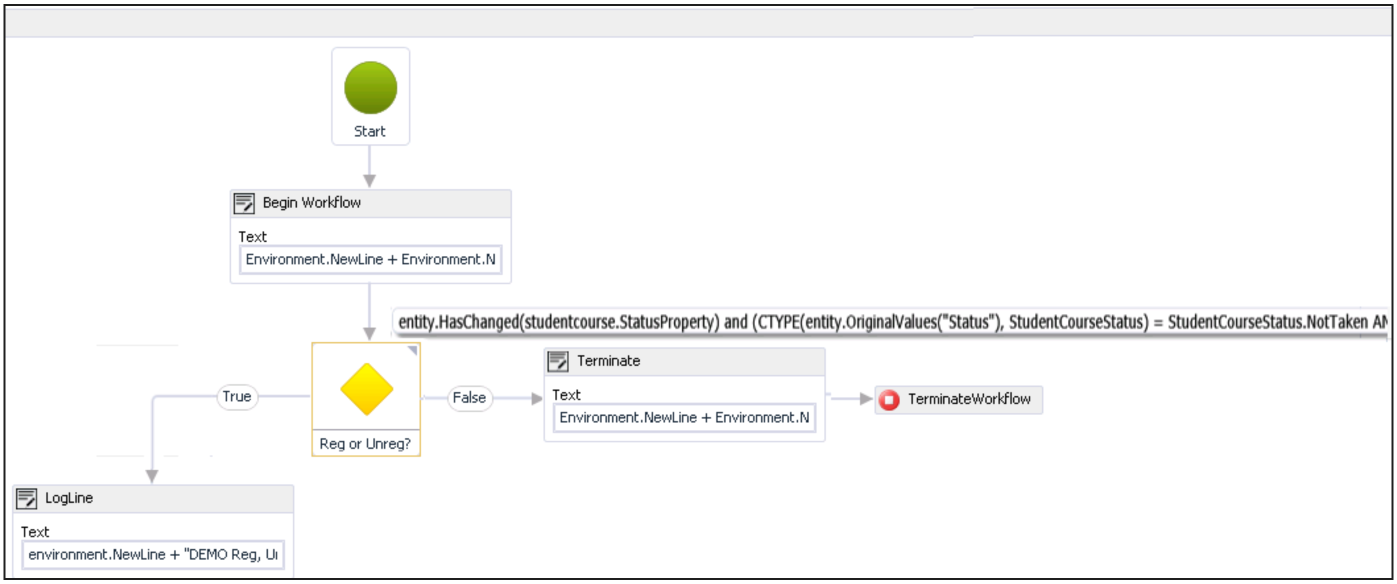
Note: If you are updating legacy activities to the new object model, be sure to update the variable type for validation messages. Many of the legacy activities use the variable type 'ValidationMessage', while the new object model uses the variable type ' ValidationMessageCollection'. It is not enough to create a variable in the new object model, you also need to instantiate the variable.

Copy/Paste Sequences

If you copy and paste a Sequence from one workflow to another, you may need to recreate any associated variables to ensure all namespaces are properly imported.

Check for StudentCourse.Status Changes

If you are using the Status property in workflows that check for StudentCourse.Status changes, use a logic pattern containing the CTYPE function with multiple combinations of possible status changes.



entity.HasChanged(studentcourse.StatusProperty) and (CTYPE(entity.OriginalValues("Status"), StudentCourseStatus) = StudentCourseStatus.NotTaken AN

Start
Begin Workflow
Text
Environment.NewLine + Environment.N
True
False
Reg or Unreg?
Terminate
Text
Environment.NewLine + Environment.N
TerminateWorkflow
LogLine
Text
environment.NewLine + "DEMO Reg, Ur

In our example, the FlowDecision activity contains a condition that checks whether the StudentCourse.StatusProperty entity has changed and whether the original Status value was NotTaken (case a), Registered (case b), or CurrentlyAttending (case c). The CTYPE function changes the original Status values to a new Status values for each case.

entity.HasChanged(studentcourse.StatusProperty)

AND (CTYPE(entity.OriginalValues("Status"), StudentCourseStatus) = StudentCourseStatus.**NotTaken**

AND entity.Status = StudentCourseStatus.**Registered**)

OR (CTYPE(entity.OriginalValues("Status"), StudentCourseStatus) = StudentCourseStatus.**Registered**

AND entity.Status = StudentCourseStatus.**NotTaken**)

OR (CTYPE(entity.OriginalValues("Status"), StudentCourseStatus) = StudentCourseStatus.**CurrentlyAttending**

AND entity.Status = StudentCourseStatus.**Withdrawal**)

For different Status changes, replace the Status values as shown in the following pattern:

Where:

- `status1a` = original status (case a)
- `status2a` = new status (case a)
- `status1b` = original status (case b)
- `status2b` = new status (case b)
- `status1c` = original status (case c)
- `status2c` = new status (case c)

entity.HasChanged(studentcourse.StatusProperty)

AND (CTYPE(entity.OriginalValues("Status"), StudentCourseStatus) = StudentCourseStatus.**status1a**

AND entity.Status = StudentCourseStatus.**status2a**)

OR (CTYPE(entity.OriginalValues("Status"), StudentCourseStatus) = StudentCourseStatus.**status1b**

AND entity.Status = StudentCourseStatus.**status2b**)

OR (CTYPE(entity.OriginalValues("Status"), StudentCourseStatus) = StudentCourseStatus.**status1c**

AND entity.Status = StudentCourseStatus.**status2c**)

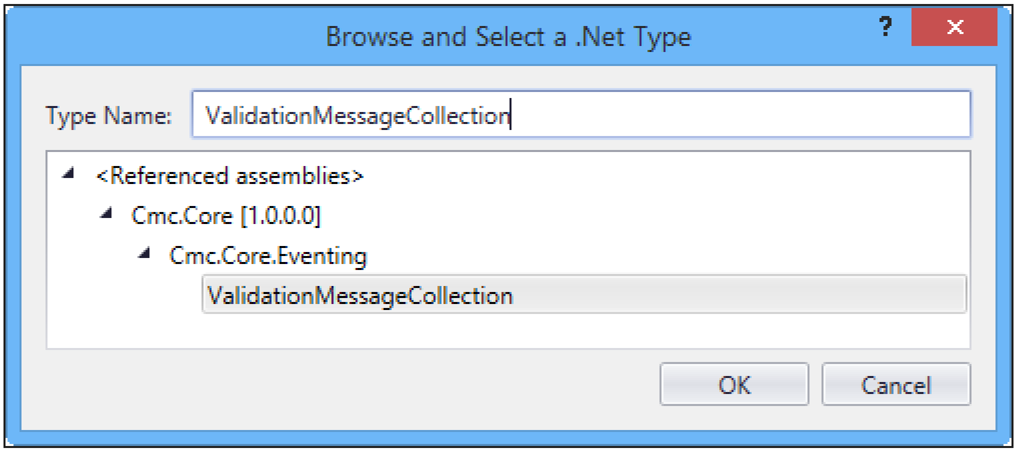
Improve Search Performance on "Browse for Types..."

When you need to select the **Browse for Types...** option in Workflow Composer, the search performance is improved if you copy and paste the entirety of the type to be searched into the "Browse and Select a .Net Type" window.

Example

You need to browse for a variable type named "ValidationMessageCollection". The quickest way to locate the variable type is:

1. Open **Notepad**.
2. Type `ValidationMessageCollection`.
3. Copy/paste `ValidationMessageCollection` into **Type Name** field of the "Browse and Select a .Net Type" window.



Browse and Select a .Net Type

Type Name: ValidationMessageCollection

<Referenced assemblies>
Cmc.Core [1.0.0.0]
Cmc.Core.Eventing
ValidationMessageCollection

How to Initialize an Array

You can initialize an array in an Assign activity.

Examples

- Boolean array:

  ```
  New Boolean() {false, false}
  ```

  — OR —

  ```
  {false,false}
  ```

- Integer array

  ```
  New Integer() {1, 2, 4, 8}
  ```

- Nested array

  ```
  {{1,2},{3,4}}
  ```

You don’t have to worry about the size of the array. The number of values it will have defines the size.

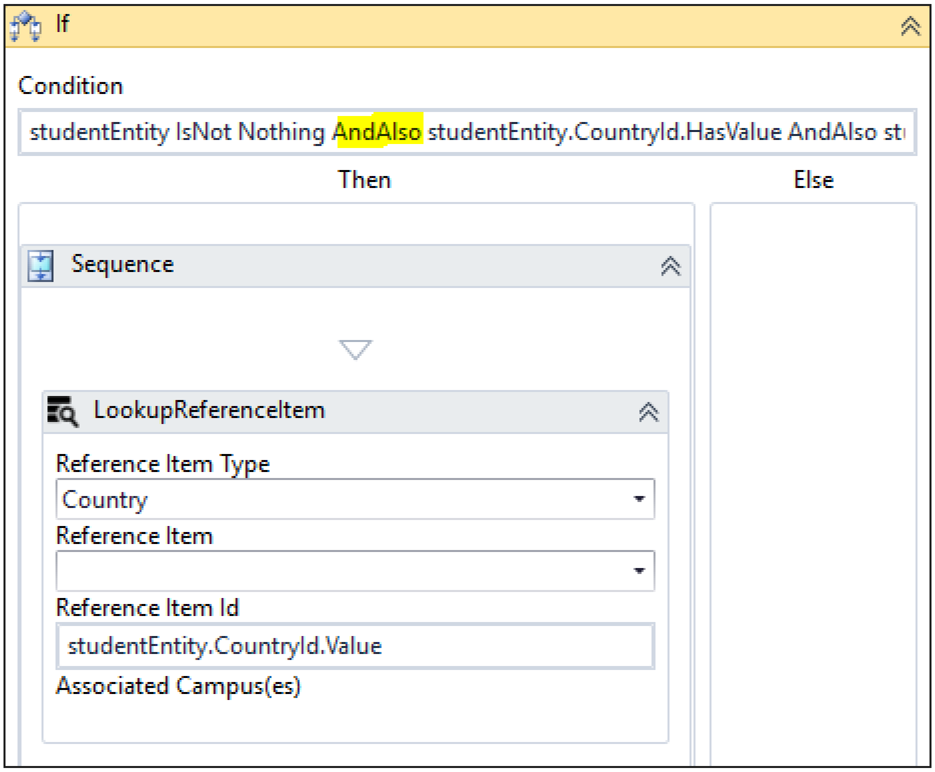
To access these array elements, note that the index always starts at 0.

AndAlso Operator

You can combine expressions using operators. The **And** operator evaluates expressions on both sides. The **AndAlso** operator evaluates the right side if and only if the left side is true. The right way of exiting the evaluation (and preventing "Object reference not set to instance" errors) is to use AndAlso.

Example

studentEntity IsNot Nothing **AndAlso** studentEntity.CountryId.HasValue **AndAlso** studentEntity.CountryId.Value > 0



Condition: studentEntity IsNot Nothing AndAlso studentEntity.CountryId.HasValue AndAlso st

Reference Item Type: Country

Reference Item:

Reference Item Id: studentEntity.CountryId.Value

Associated Campus(es)

Host Processes

The hosts involved in the workflow vary depending on the Anthology configuration and environment. The ServiceModuleHost, ServiceBrokerServiceModule, and the WorkflowServiceModule are required to host workflow processes. Application servers and clients vary.

Host Process	Description
ServiceModuleHost.exe	Windows service responsible for hosting plugin modules to simplify deployment and maintenance of processes that run in the background. For more information, see Service Module Host.
ServiceBrokerServiceModule	Responsible for monitoring SQL Server Service Broker Queues for messages. Currently, message handlers are implemented to raise EventService events and trigger schedule-based workflows.
WorkflowServiceModule	Responsible for executing runnable workflows that have been persisted using the Delay activity. This process waits for suspended workflows (a.k.a. long running) to resume. It queries the database every 10 seconds. This process waits for suspended workflows (a.k.a. long running) to resume. It queries the database every 10 seconds. It requires a valid handle in the database to ensure that the process is valid and connected to the database. The process refreshes a lock within the database table: `[System.Activities.DurableInstancing].[LockOwnersTable]` every 30 seconds. If the lock becomes expired or if it is not found, the module will start to throw exceptions regarding the lock being Freed or Invalid.
CampusVue.exe	Desktop Client for Anthology Student
w3wp.exe	IIS hosted application server. Events are raised through ASP.NET or WCF.
WorkflowComposer.exe	Allows power users to create and publish workflows and track workflow instances.

API Authentication for Workflow Activities

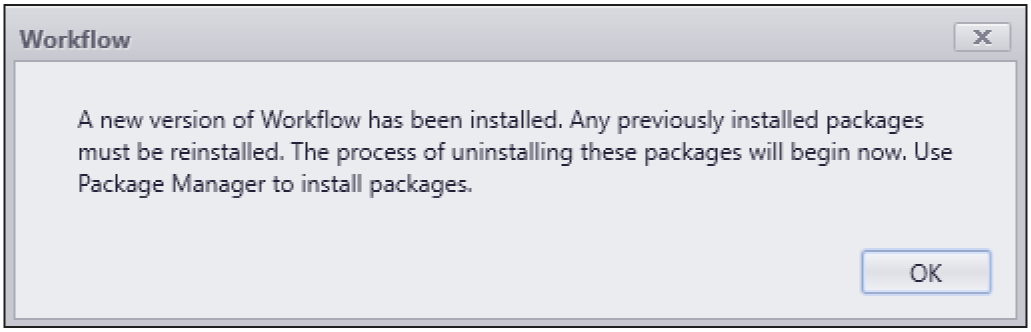
Installation Manager accepts the API Username and Password to allow applications other than Anthology Student to execute Anthology Student workflow activities. The API Username and Password values are specified on the Anthology Student tab in the Forms Builder Settings screen of Installation Manager. The API Username and Password are written to the SyRegistry table within the Anthology Student database (with selected encryption mechanism). The API Username and Password are no longer written to the web.config file and to the app config of the Service Module Host for Workflow Composer.

Package Manager

The Package Manager application is integrated in Workflow Composer. Package Manager displays workflow packages accessible by the configured customer. The packages contain contracts, entities, events, and activities related to workflows and eventing for CampusNexus CRM, Anthology Student, and Forms Builder. The packages must be installed before you can start creating workflows.

Note: If you installed Workflow Composer using ClickOnce with auto update, previously installed packages are removed and need to be reinstalled.

When a new version of Workflow Composer has been installed, the following message will remind you to reinstall any packages.



A new version of Workflow has been installed. Any previously installed packages must be reinstalled. The process of uninstalling these packages will begin now. Use Package Manager to install packages.

Click **OK** and proceed to install the needed Activities and Contracts .msi packages using Package Manager.

For each .msi package that you install, you will be prompted to confirm that you want to allow the app to make changes to your device.

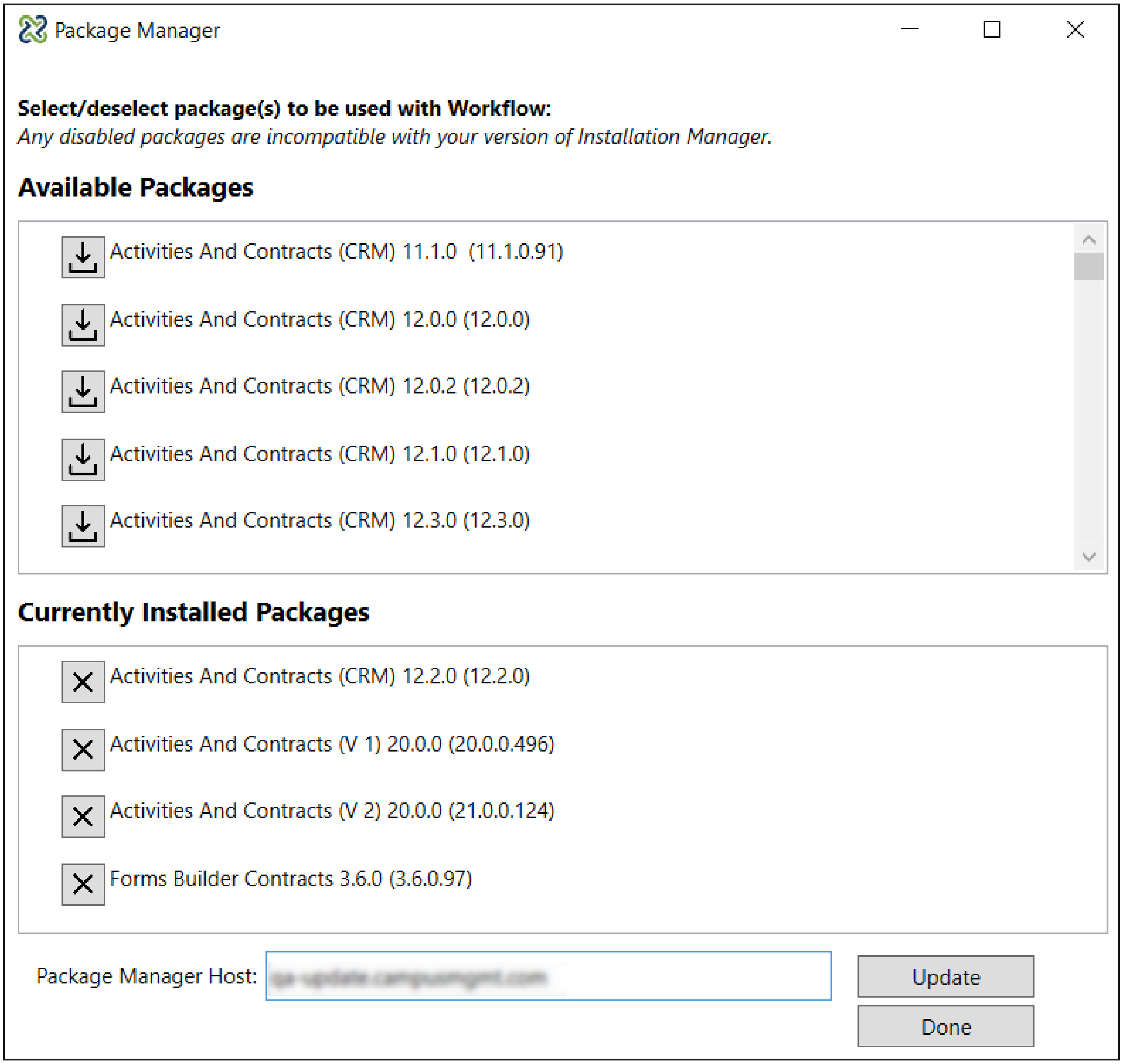
Install Packages

1. Right-click the Workflow Composer icon on your desktop and select **Run as administrator**.
2. Click **Package Manager** in the toolbar of Workflow Composer. Because Workflow Composer cannot be updated while it is running, Package Manager prompts you to close the Workflow Composer.

   ⚠ Depending on the settings and antivirus/malware software installed on your machine as well as your corporate policies, you may see a warning when installing Workflow Composer and its activity packages.

3. Click **Yes** to proceed. The Package Manager window is displayed.

   Note: Check the URL of the **Package Manager Host** for your environment. If necessary, change the URL and click **Update** before trying to install packages.



Package Manager

Select/deselect package(s) to be used with Workflow:
Any disabled packages are incompatible with your version of Installation Manager.

Available Packages

- ⤓ Activities And Contracts (CRM) 11.1.0 (11.1.0.91)
- ⤓ Activities And Contracts (CRM) 12.0.0 (12.0.0)
- ⤓ Activities And Contracts (CRM) 12.0.2 (12.0.2)
- ⤓ Activities And Contracts (CRM) 12.1.0 (12.1.0)
- ⤓ Activities And Contracts (CRM) 12.3.0 (12.3.0)

Currently Installed Packages

- ☒ Activities And Contracts (CRM) 12.2.0 (12.2.0)
- ☒ Activities And Contracts (V 1) 20.0.0 (20.0.0.496)
- ☒ Activities And Contracts (V 2) 20.0.0 (21.0.0.124)
- ☒ Forms Builder Contracts 3.6.0 (3.6.0.97)

Package Manager Host: [redacted]

4. In the **Available Packages** pane, click ⤓ for the package to install. A progress bar displayed while the selected package is being downloaded and extracted to the appropriate location. When the installation is complete, click **Done** to close Package Manager.

   You can install only one version of a specific package type. For example, if you installed "Activities and Contracts (CRM) 12.0.0", you cannot have "Activities and Contracts (CRM) 13.0.0" on the same instance of Workflow Composer at the same time. "Activities and Contracts (CRM) 13.0.0" will overwrite "Activities and Contracts (CRM) 12.0.0".

   For each version of Anthology Student, Package Manager provides Activities and Contracts for the legacy namespaces and the new namespaces. The Activities and Contracts packages for legacy namespaces are labeled **V1**, while the Activities and Contracts for new namespaces are labeled **V2**. For more information, see About the New Object Model.

Note: Anthology Student 21.0 (and later) Activities and Contracts are required when using Workflow Composer with Web API connection. Earlier versions of Activities and Contracts are incompatible.

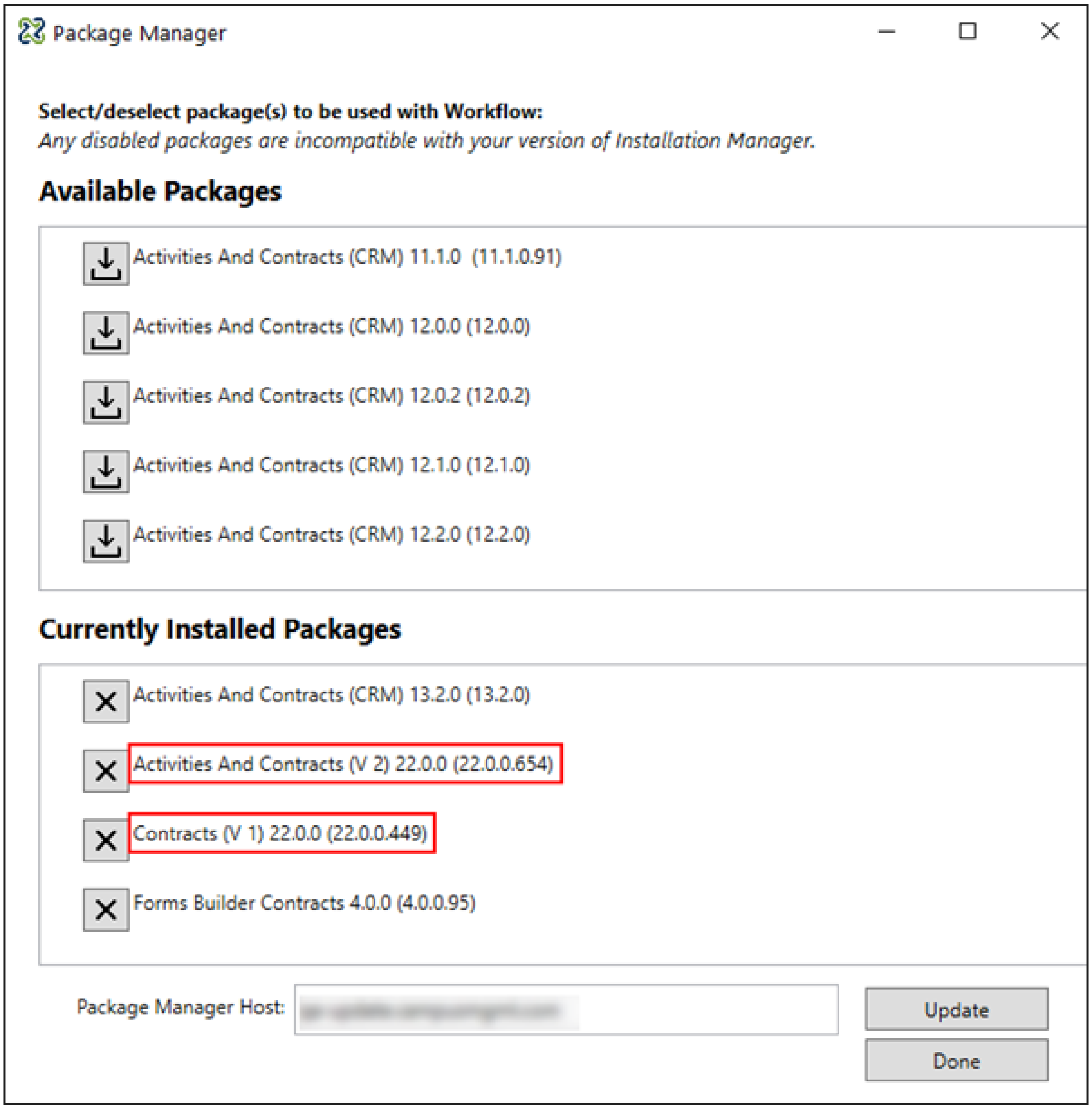
With Workflow Composer 4.x and Anthology Student 22.x0 and later, you need to install both the V2 Activities and Contracts and the V1 Contract packages. V1 Activities are not supported in 22.x and later.

If you have workflows with V1 Activities, warning messages will be displayed when you select or run a workflow. See Run Time Messages About V1 Activities.

If workflows that contain V1 Activities have not been updated prior to upgrading to Anthology Student 22.x and installing 22.x Activities and Contracts packages, perform the following steps:

1. Uninstall the V1 and V2 packages for 22.x.
2. Import an earlier version of V1 and V2 packages (e.g., 21.x).
3. Update the workflows to replace the V1 activities (see Actions Required).
4. Re-import the 22.x packages.

The packages for Anthology Student 22.x and later will only contain the V1 Contracts and not the Activities.



Select/deselect package(s) to be used with Workflow:
Any disabled packages are incompatible with your version of Installation Manager.

Available Packages

- Activities And Contracts (CRM) 11.1.0 (11.1.0.91)
- Activities And Contracts (CRM) 12.0.0 (12.0.0)
- Activities And Contracts (CRM) 12.0.2 (12.0.2)
- Activities And Contracts (CRM) 12.1.0 (12.1.0)
- Activities And Contracts (CRM) 12.2.0 (12.2.0)

Currently Installed Packages

- Activities And Contracts (CRM) 13.2.0 (13.2.0)
- Activities And Contracts (V 2) 22.0.0 (22.0.0.654)
- Contracts (V 1) 22.0.0 (22.0.0.449)
- Forms Builder Contracts 4.0.0 (4.0.0.95)

Package Manager Host: [redacted]

5. Restart Workflow Composer. The contracts, entities, events, and activities associated with the downloaded packages are now available in Workflow Composer.

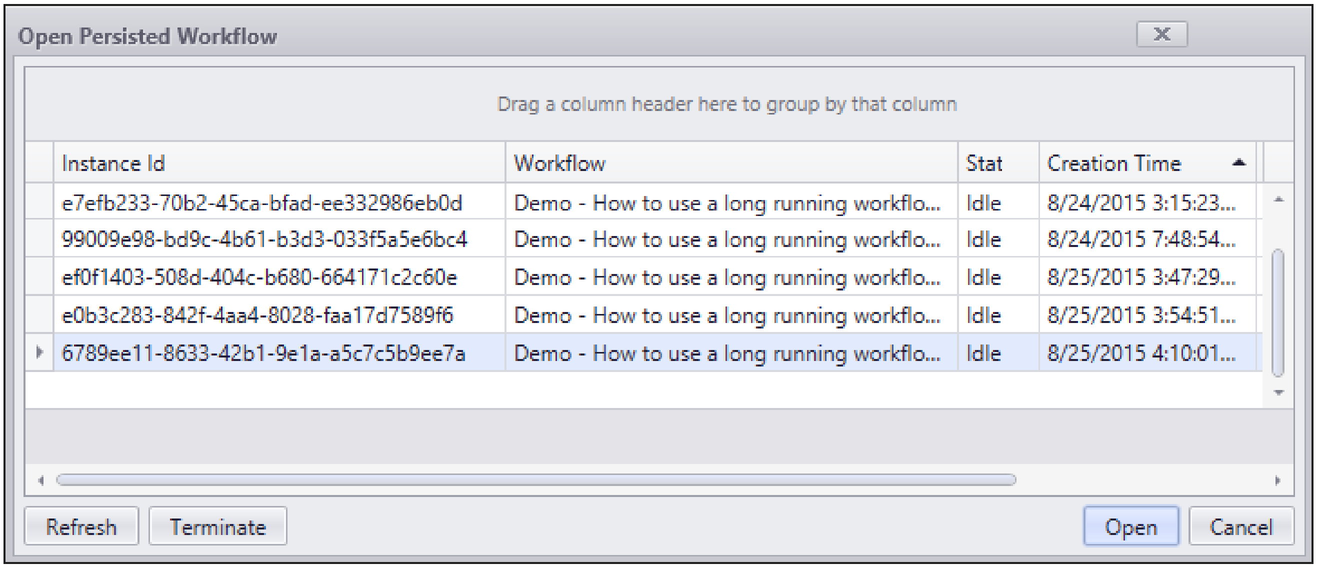
Uninstall Packages

1. Right-click the Workflow Composer icon on your desktop and select **Run as administrator**.
2. Click **Package Manager** in the toolbar of Workflow Composer.
3. Click **Yes** to close Workflow Composer. Package Manager displays check marks for any previously installed packages.
4. In the **Currently Installed Packages** pane, click ☒ for the package to uninstall. A progress bar displayed while the selected package is being removed. Click **Done** to close Package Manager.
5. Restart Workflow Composer. The uninstalled packages are no longer available.

Persisted Workflows

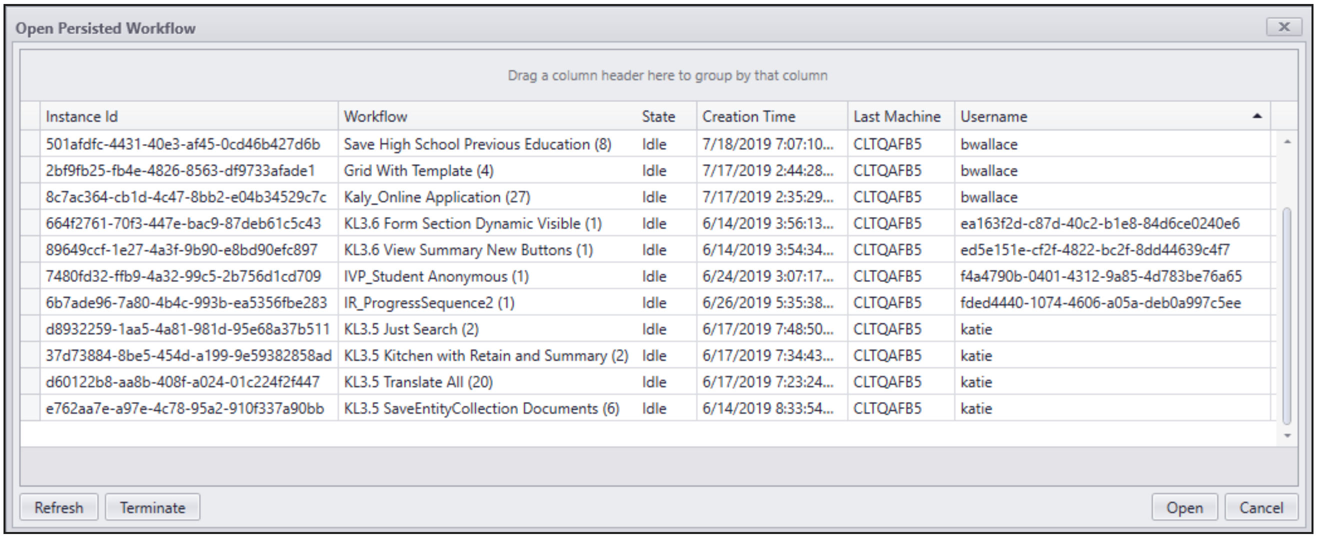
The Workflow application enables you to open, refresh, and terminate persisted workflows. Persisted workflows may contain Delay or Bookmark activities or are started by a Scheduled event. These workflows reside in the database and are idle until the delay, bookmark, or scheduled events occur.

1. In Workflow Composer on the Home tab of the ribbon, click **Open Persisted Workflow** .
2. The Open Persisted Workflow window is displayed. You can sort and filter the grid as needed.



Instance Id	Workflow	Stat	Creation Time
e7efb233-70b2-45ca-bfad-ee332986eb0d	Demo - How to use a long running workflo...	Idle	8/24/2015 3:15:23...
99009e98-bd9c-4b61-b3d3-033f5a5e6bc4	Demo - How to use a long running workflo...	Idle	8/24/2015 7:48:54...
ef0f1403-508d-404c-b680-664171c2c60e	Demo - How to use a long running workflo...	Idle	8/25/2015 3:47:29...
e0b3c283-842f-4aa4-8028-faa17d7589f6	Demo - How to use a long running workflo...	Idle	8/25/2015 3:54:51...
6789ee11-8633-42b1-9e1a-a5c7c5b9ee7a	Demo - How to use a long running workflo...	Idle	8/25/2015 4:10:01...

In Workflow Composer 3.0 and later, the Persisted Workflow grid has an additional "Username" column. This column is populated only for workflows associated with Forms Builder sequences.



Instance Id	Workflow	State	Creation Time	Last Machine	Username
501afdfc-4431-40e3-af45-0cd46b427d6b	Save High School Previous Education (8)	Idle	7/18/2019 7:07:10...	CLTQAFB5	bwallace
2bf9fb25-fb4e-4826-8563-df9733afade1	Grid With Template (4)	Idle	7/17/2019 2:44:28...	CLTQAFB5	bwallace
8c7ac364-cb1d-4c47-8bb2-e04b34529c7c	Kaly_Online Application (27)	Idle	7/17/2019 2:35:29...	CLTQAFB5	bwallace
664f2761-70f3-447e-bac9-87deb61c5c43	KL3.6 Form Section Dynamic Visible (1)	Idle	6/14/2019 3:56:13...	CLTQAFB5	ea163f2d-c87d-40c2-b1e8-84d6ce0240e6
89649ccf-1e27-4a3f-9b90-e8bd90efc897	KL3.6 View Summary New Buttons (1)	Idle	6/14/2019 3:54:34...	CLTQAFB5	ed5e151e-cf2f-4822-bc2f-8dd44639c4f7
7480fd32-ffb9-4a32-99c5-2b756d1cd709	IVP_Student Anonymous (1)	Idle	6/24/2019 3:07:17...	CLTQAFB5	f4a4790b-0401-4312-9a85-4d783be76a65
6b7ade96-7a80-4b4c-993b-ea5356fbe283	IR_ProgressSequence2 (1)	Idle	6/26/2019 5:35:38...	CLTQAFB5	fded4440-1074-4606-a05a-deb0a997c5ee
d8932259-1aa5-4a81-981d-95e68a37b511	KL3.5 Just Search (2)	Idle	6/17/2019 7:48:50...	CLTQAFB5	katie
37d73884-8be5-454d-a199-9e59382858ad	KL3.5 Kitchen with Retain and Summary (2)	Idle	6/17/2019 7:34:43...	CLTQAFB5	katie
d60122b8-aa8b-408f-a024-01c224f2f447	KL3.5 Translate All (20)	Idle	6/17/2019 7:23:24...	CLTQAFB5	katie
e762aa7e-a97e-4c78-95a2-910f337a90bb	KL3.5 SaveEntityCollection Documents (6)	Idle	6/14/2019 8:33:54...	CLTQAFB5	katie

3. Select a workflow instance.
4. Click **Open** to view the workflow definition. You can edit and save the workflow.

⚠ The process retrieves and displays data from durable instancing (not tracking); However, if a record is selected and the Open button is clicked, the process attempts to retrieve tracking data. If the tracking database is not configured, the process will continue without error and will still open the persisted workflow.

5. Click **Refresh** to update the grid of persisted workflows.
6. Click **Terminate** to stop a workflow. Click **Yes** to confirm. The workflow instance is removed from the grid.
7. Click **Cancel** to close the Open Persisted Workflow window.

Note:

Workflow tracking relies on three database strings that are configured in the configuration file for the ServiceModuleHost.exe. For more information, see Connection Strings.

a. `dbConnection`
b. `WorkflowDurableInstancingConnection`
c. `WorkflowTrackingConnection`

The `dbConnection` and `WorkflowDurableInstancingConnection` should point to the same SIS database, e.g., a Anthology Student database. The `WorkflowTrackingConnection` should point to a specific tracking database (different than the SIS database).

Save and Publish Workflows

The Workflow application enables you to save a local copy of a workflow and publish it when it is ready to be run by the workflow engine.

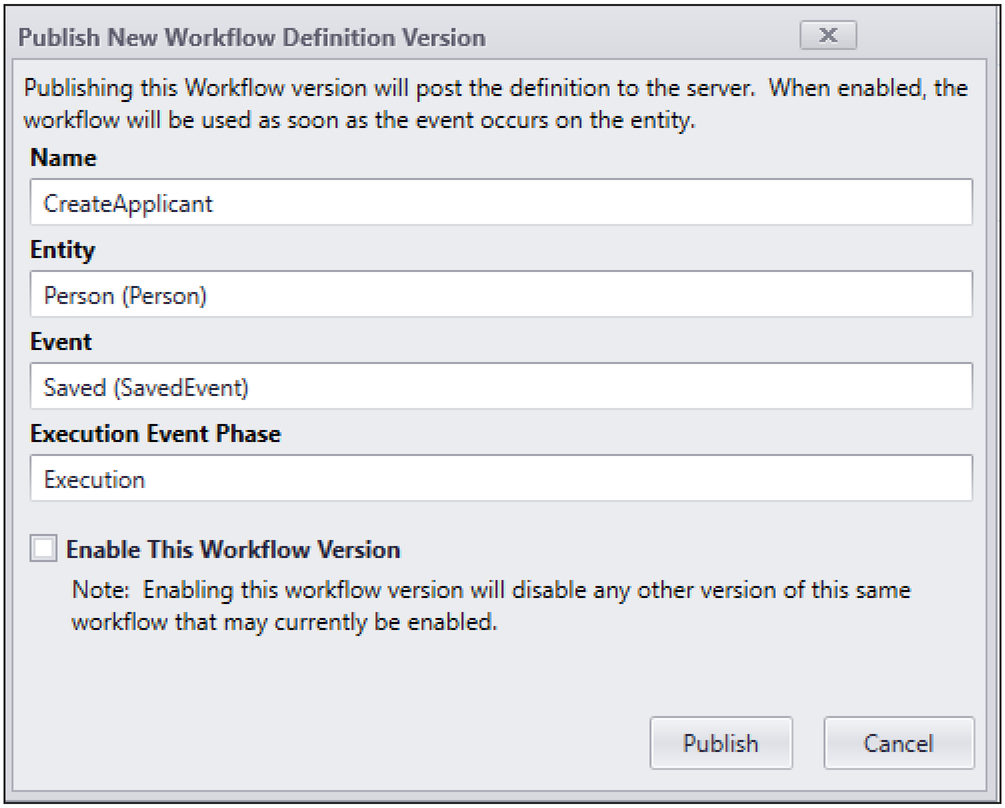
The option to save to the local file system is intended to be used during the design phase and for file sharing purposes. Workflows that are stored locally are not used by the workflow engine.

To save a workflow locally, click **Save** or **Save As...** on the Home tab of the Designer.

Workflows that are ready to be run by the workflow engine must be published. Published workflows are stored in the database. During publishing, you have the option to enable the workflow. The workflow engine runs only workflows that are published and enabled. Multiple versions of a workflow can be saved to the database. If another workflow with same Entity.Event and Name is found, the publishing process increments the workflow version. Only one version of a particular workflow can be enabled at a time.

1. Open a workflow definition in Workflow Composer. See View, Enable, and Delete Workflows.
2. On the Home tab, click **Publish**. The "Publish New Workflow Definition Version" window is displayed.

   The **Name**, **Entity**, **Event**, and **Execution Event Phase** fields are automatically populated based on the information that was gathered when the workflow definition was created. For more information about Event Phases, see Create Workflows with Event Phase.



Publish New Workflow Definition Version

Publishing this Workflow version will post the definition to the server. When enabled, the workflow will be used as soon as the event occurs on the entity.

Name: CreateApplicant

Entity: Person (Person)

Event: Saved (SavedEvent)

Execution Event Phase: Execution

□ Enable This Workflow Version

Note: Enabling this workflow version will disable any other version of this same workflow that may currently be enabled.

3. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.

   Enabling the workflow disables any other version of the same workflow that may be currently enabled.

4. Click **Publish**.

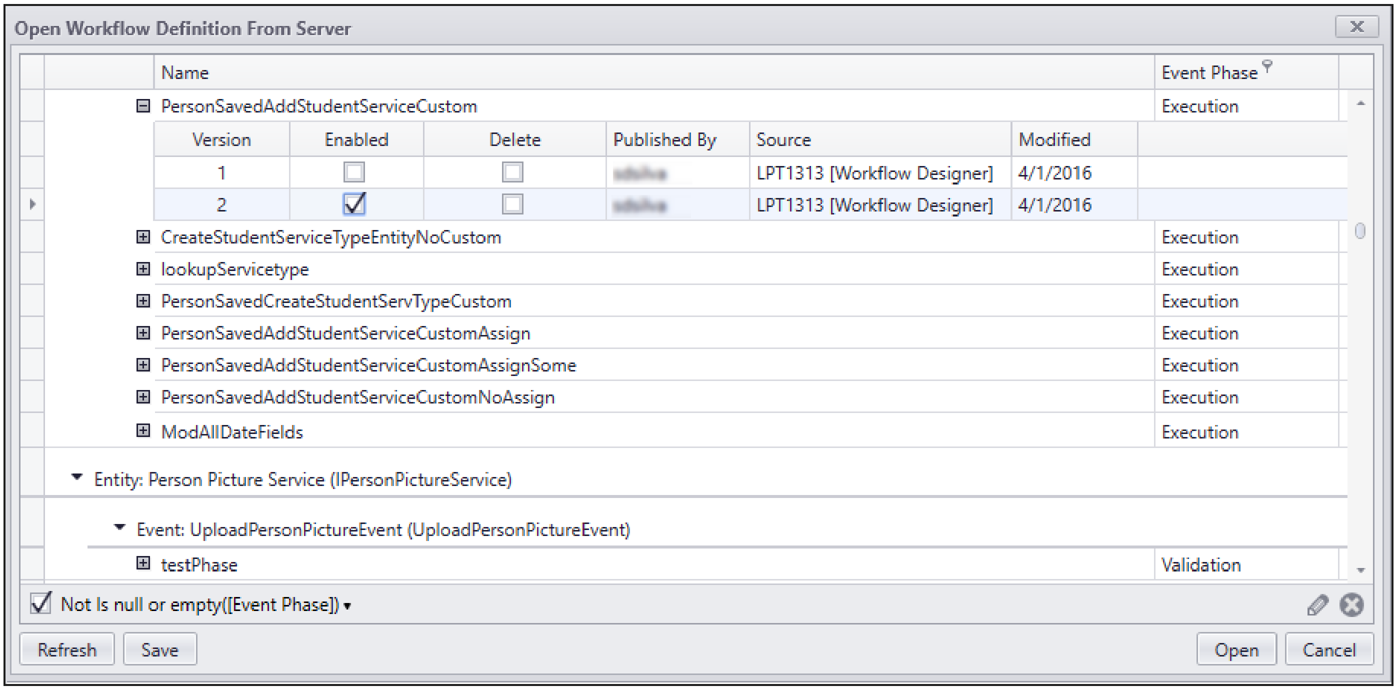
View, Enable, and Delete Workflows

Workflow Composer enables you to open and view workflow definitions from a local file system or from an SQL Server database table. Workflows stored in the database can be enabled and disabled.

View Workflows from File or Server

To open a workflow from the file system, click **Open** in the **File** section of the ribbon and navigate to the location where your workflow files are stored.

To open a workflow from the database, click **Open** in the **Server** section of the ribbon. The "Open Workflow Definition From Server" window displays a grid with information about workflows that have been published to the database. To load a workflow definition into the Designer, select the workflow in the grid and click **Open**.



Open Workflow Definition From Server

Name	Event Phase
⊟ PersonSavedAddStudentServiceCustom	Execution

Version	Enabled	Delete	Published By	Source	Modified
1	□	□	[redacted]	LPT1313 [Workflow Designer]	4/1/2016
2	☑	□	[redacted]	LPT1313 [Workflow Designer]	4/1/2016

Name	Event Phase
⊞ CreateStudentServiceTypeEntityNoCustom	Execution
⊞ lookupServicetype	Execution
⊞ PersonSavedCreateStudentServTypeCustom	Execution
⊞ PersonSavedAddStudentServiceCustomAssign	Execution
⊞ PersonSavedAddStudentServiceCustomAssignSome	Execution
⊞ PersonSavedAddStudentServiceCustomNoAssign	Execution
⊞ ModAllDateFields	Execution

Entity: Person Picture Service (IPersonPictureService)

Event: UploadPersonPictureEvent (UploadPersonPictureEvent)

Name	Event Phase
⊞ testPhase	Validation

☑ Not Is null or empty([Event Phase])

Workflows are grouped by entities and events in the "Open Workflow Definition From Server" window. Expand the entity and event groups to view the following information about each workflow:

- Version
- Enabled
- Delete
- Published by (Windows identity of the user who is signed in to the Workflow Composer workstation
- Source (Windows identity of the workstation where the workflow came from)
- Date modified
- Event Phase

You can sort the grid by workflow **Name** and filter by **Event Phase**. For more information about Event Phases, see Create Workflows with Event Phase.

You can also manipulate the grid in the Open Workflow Definition From Server window. Hover over the column headings to access column filter and sort controls. Drag column headers to rearrange or remove columns.

Enable a Workflow

Select or clear the **Enabled** check box to choose which workflow is currently active and will be selected by the workflow engine to execute when a new instance of this workflow is invoked. For workflows that have multiple versions, only one version at a time can be enabled. Click **Save** when you have changed the enabled status.

Workflow Versioning

When you open, modify, and publish a workflow version, the version number of the workflow is automatically incremented, and the new version is added to the grid.

You can modify long running workflows when needed without disrupting any instances of the workflow that are in process and persisted to the data store. The execution of any currently persisted workflows is completed using the definition of the older version and invokes new instances of the workflow using the latest definition.

The WorkflowIdentity class supports the versioning and dynamic update functionality of Workflows and enables hosting multiple versions of the same workflow.

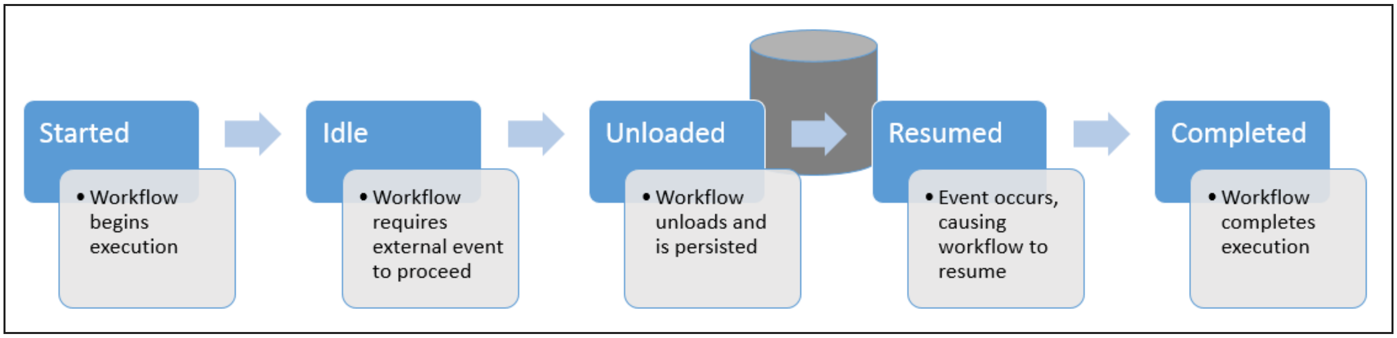
Delete Workflow Definitions

The 'Open Workflow Definition from Server' window enables you to delete workflow definitions that are stored in the database. You can select multiple revisions of a single workflow, all revisions of workflow, and workflow revisions of multiple different contracts at same time. When you select the **Delete** check box, you are prompted to confirm that you want to permanently delete the selected workflows/revisions.

If at least one instance of any of the selected workflow revisions is a long running workflow and still is in process, a message states that one or more instances of one of the selected workflow revisions is still in process. If you proceed with the Delete operation, all in process instances of workflows associated with any of the selected workflow revisions are deleted as well as the selected workflow revisions.

Workflow Execution Scenarios

A workflow continuously executes activities until there are no more activities to execute or until all currently executing activities are waiting for input. The input can come from a user, an external system, or an expiring timer. While waiting for input, the workflow becomes idle. A host can unload workflows that have gone idle and reload them to continue execution when the input arrives. To unload the workflow when it becomes idle, the host must persist the workflow instance.


Started
• Workflow begins execution
Idle
• Workflow requires external event to proceed
Unloaded
• Workflow unloads and is persisted
Resumed
• Event occurs, causing workflow to resume
Completed
• Workflow completes execution

Persistence of the workflow instances and associated data is required in the following scenarios:

- A workflow is started within an application, unloaded due to a Bookmark, and resumed from the same application.
- A workflow is started within an application server, unloaded due to a Delay, and resumed from the application server.
- A workflow is started based on a Schedule, unloaded due to a Delay, and resumed from the application server.
- A workflow is started based on a Schedule, unloaded due to a Bookmark, and resumed from the application server.

The hosts involved in the workflow vary depending on the Anthology configuration and environment. For more information about the hosts, see Host Processes.

Bookmark

Bookmarks are the mechanism that enables a workflow activity to passively wait for input without holding onto a workflow thread. A bookmark is the point at which execution can be resumed (and through which input can be delivered) within a workflow instance. External code is responsible for resuming the bookmark with relevant data. Multiple bookmarks can be scheduled at the same time.

For information about creating different bookmark types, see CreateBookmark and CreateBookmark<>.

To see how CreateBookmark and ResumeBookmark activities can be used in a workflow, refer to:

- Create a Long Running Workflow
- Wake up the Long Running Workflow

Delay

A Delay activity creates a timer for a specified duration. The workflow instance is unloaded until the timer expires.

Other activities related to workflow persistence include:

- StateMachine
- State
- FinalSate
- Persist
- NoPersistScope
- Pick
- PickBranch
- Parallel

For more information about these activities, see Generic Activities.

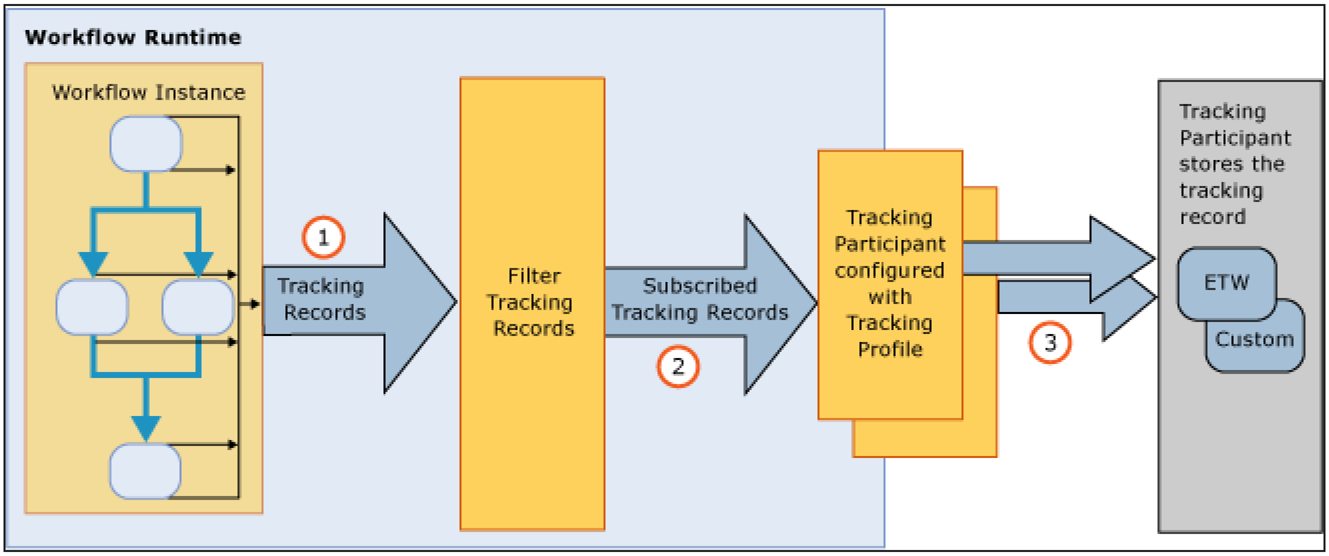
Schedule

Event scheduling can be used to start a workflow on a recurring schedule. For more information see Event Scheduling.

The web client for Anthology Student 20.0 and later enables you to set up schedules to trigger workflows. In the web client, select the Processes tile, locate System Administrator in the tree, and select Background Processes. On Background Processes page, add or edit jobs and then schedule the jobs so that they are executed as a background process.

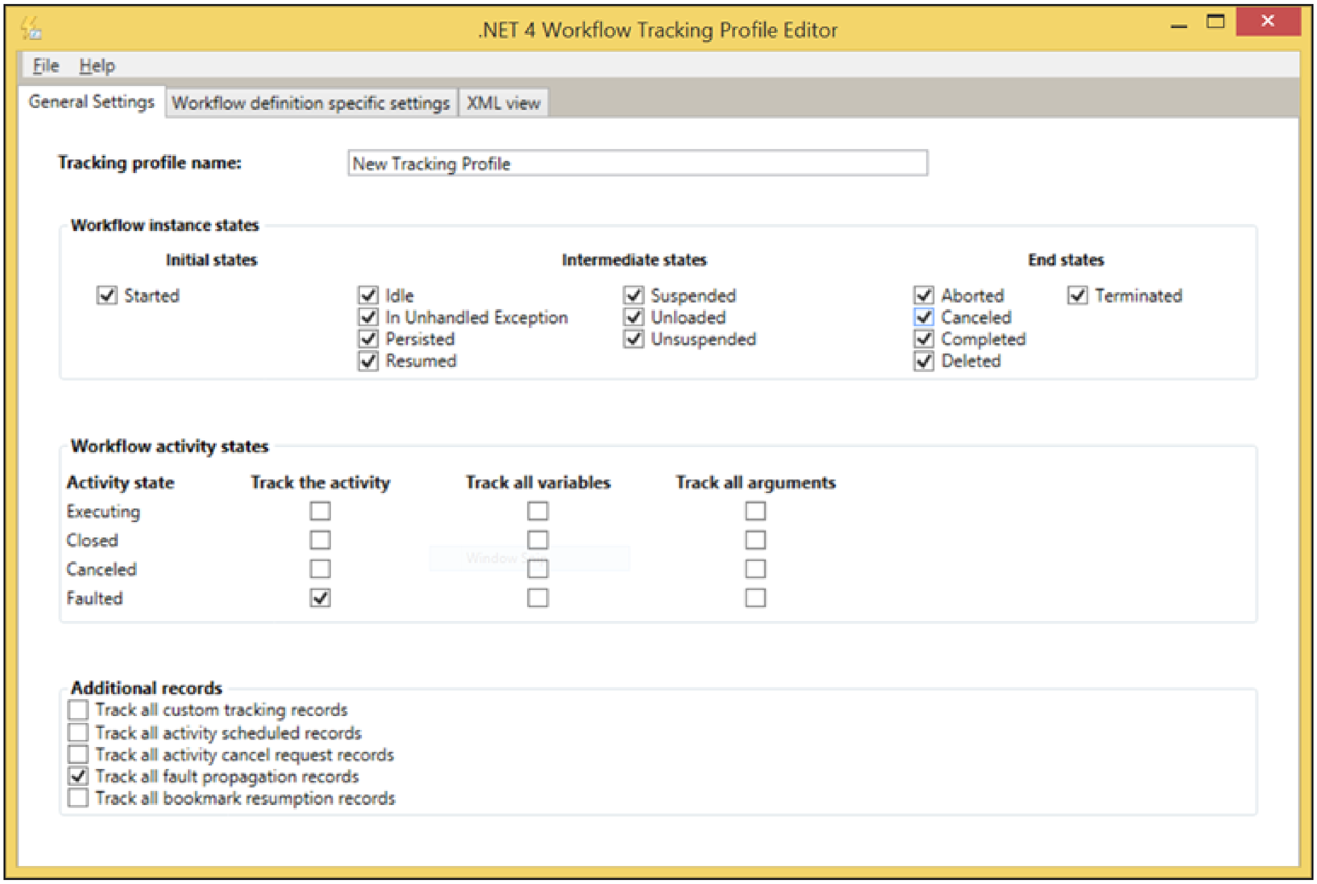
Workflow Tracking

Workflow Composer provides a visual workflow tracking feature that is built based on the workflow tracking infrastructure available in the .NET Framework. Workflow tracking enables you to observe the execution of a workflow instance at runtime.


Workflow Runtime
Workflow Instance
1
Tracking Records
Filter Tracking Records
Subscribed Tracking Records
2
Tracking Participant configured with Tracking Profile
3
Tracking Participant stores the tracking record
ETW
Custom

1. Tracking records are emitted from a workflow at the workflow instance level and when activities within the workflow execute.

2. Tracking profiles are used to specify which tracking information is emitted for a workflow instance. The queries defined within the tracking profile section define the kinds of events that are returned by the subscription. For example, a tracking profile might subscribe to Started and Completed workflow event statuses. If no profile is specified, all tracking events are emitted. Tracking profiles are XML elements within a standard .NET framework config file. A Workflow Tracking Profile Editor UI is also available.


.NET 4 Workflow Tracking Profile Editor
File
Help
General Settings
Workflow definition specific settings
XML view

Tracking profile name: New Tracking Profile

Workflow instance states

Initial states

☑ Started

Intermediate states

☑ Idle
☑ In Unhandled Exception
☑ Persisted
☑ Resumed
☑ Suspended
☑ Unloaded
☑ Unsuspended

End states

☑ Aborted
☑ Canceled
☑ Completed
☑ Deleted
☑ Terminated

Workflow activity states

Activity state	Track the activity	Track all variables	Track all arguments
Executing	☐	☐	☐
Closed	☐	☐	☐
Canceled	☐	☐	☐
Faulted	☑	☐	☐

Additional records

☐ Track all custom tracking records
☐ Track all activity scheduled records
☐ Track all activity cancel request records
☑ Track all fault propagation records
☐ Track all bookmark resumption records

3. A workflow tracking participant needs to be added to the workflow host to subscribe to tracking records. The tracking participant subscribes to TrackingRecord objects. It contains the logic to process a TrackingRecord (for example, writing to a file). The .NET Framework provides an ETW (Event Tracing for Windows) tracking participant with a basic profile that is installed in the machine.config file. Anthology also provides an SQL tracking participant that stores the tracking records and permits retrieval of the tracking records.

For more information about the workflow tracking and tracing infrastructure in .NET, see http://msdn.microsoft.com/en-us/library/ee513992(v=vs.110).aspx.

Note:

Workflow tracking relies on three database strings that are configured in the configuration file for the ServiceModuleHost.exe. For more information, see Connection Strings.

a. `dbConnection`
b. `WorkflowDurableInstancingConnection`
c. `WorkflowTrackingConnection`

The `dbConnection` and `WorkflowDurableInstancingConnection` should point to the same SIS database, e.g., an Anthology Student database. The `WorkflowTrackingConnection` should point to a specific tracking database (different than the SIS database).

The tracking process retrieves and displays data from the tracking database. If the tracking database is not configured, Workflow Composer will display a user friendly message.

Workflow Tracking Example

After you have set up your environment for workflow tracking, use Workflow Composer for visual workflow tracking. You can:

- View workflows that are currently executing.
- View workflows that executed in the past.
- Select and replay workflows.
- Refresh the display in the Current and Historical windows.

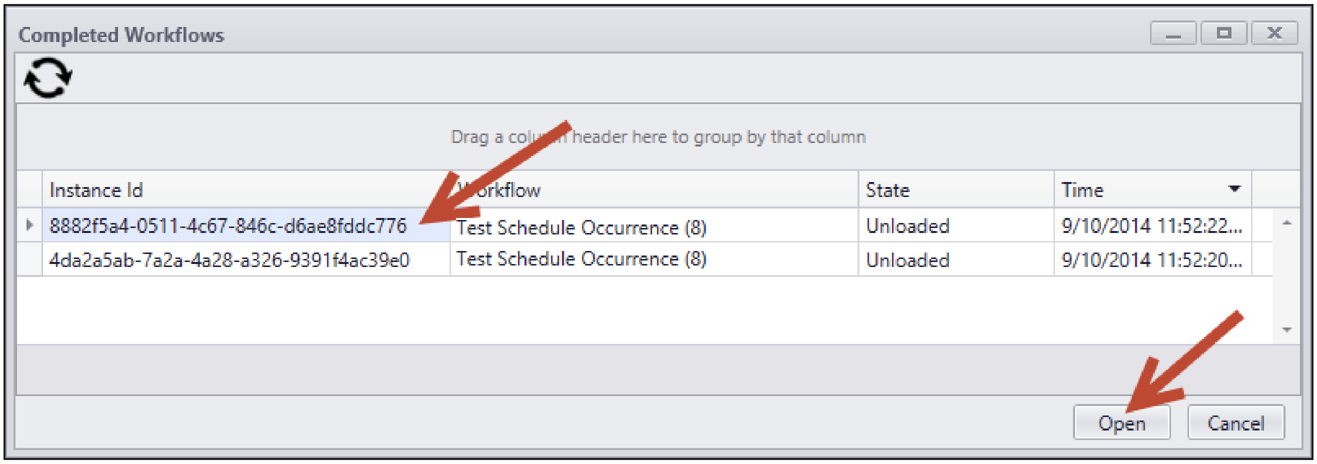
You can troubleshoot a workflow and determine if it is executing properly based on the data being passed or returned from every activity step in a given workflow.

1. Open the Workflow Designer and click **Open Tracked Workflow**.

   The Completed Workflows window is displayed. Each record indicates the following.

   - Instance Id
   - Workflow (.xaml file name)
   - State (e.g., closed, executing, idle, unloaded, completed, aborted, terminated)
   - Time

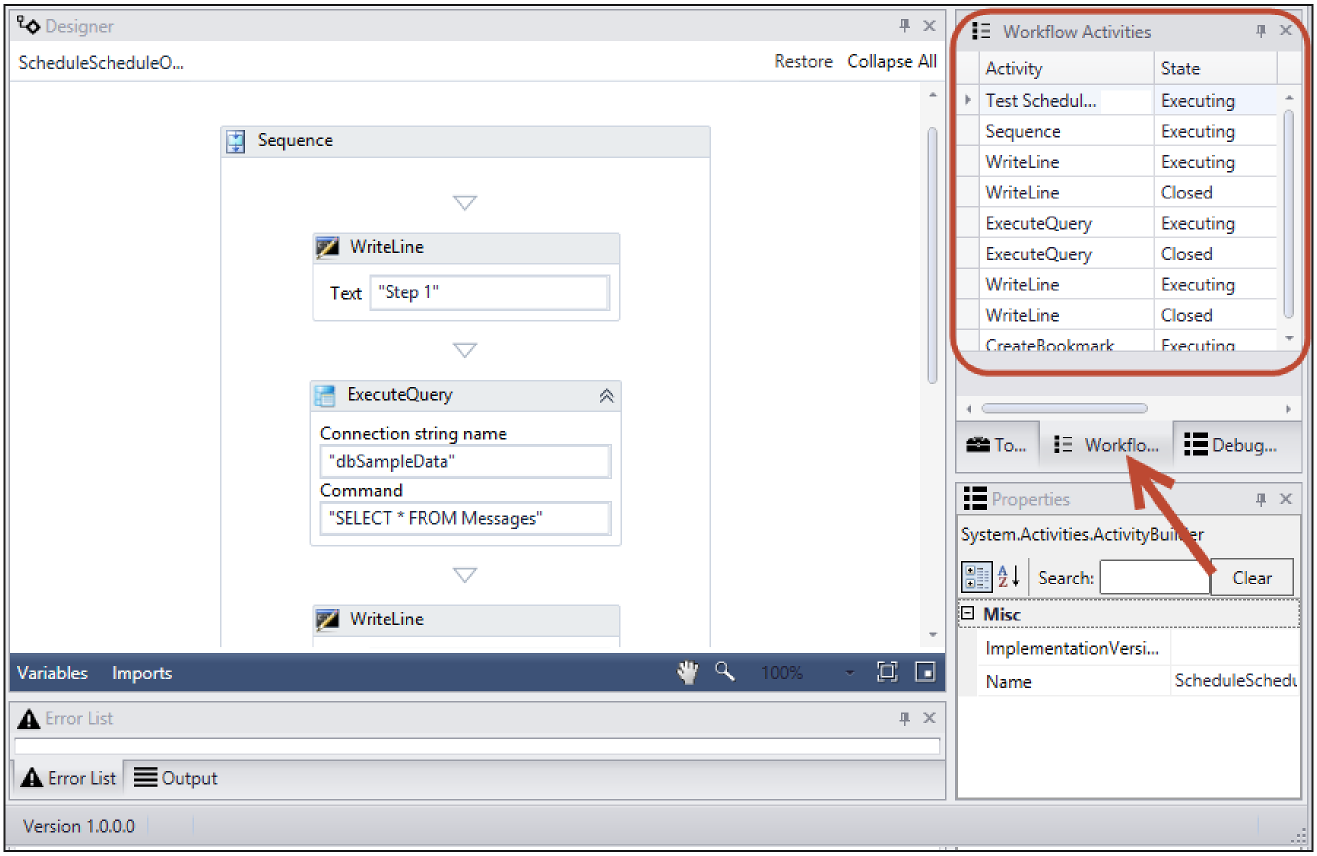
   You can sort, filter, and rearrange the columns in the grids.



Instance Id	Workflow	State	Time
8882f5a4-0511-4c67-846c-d6ae8fddc776	Test Schedule Occurrence (8)	Unloaded	9/10/2014 11:52:22...
4da2a5ab-7a2a-4a28-a326-9391f4ac39e0	Test Schedule Occurrence (8)	Unloaded	9/10/2014 11:52:20...

2. Select a record and click **Open**. The definition of the workflow instance is loaded into the Designer pane. You can select a workflow instance and click **Replay** to execute the workflow again, click **Refresh** to update the Completed Workflow instances, or click **Close** to unload the workflow from the Designer pane.

3. Click on the **Workflow Activities** tab below the Toolbox. The Workflow Activities pane is displayed. It contains records for the Activity steps in the currently loaded workflow.



Text "Step 1"

Connection string name
"dbSampleData"

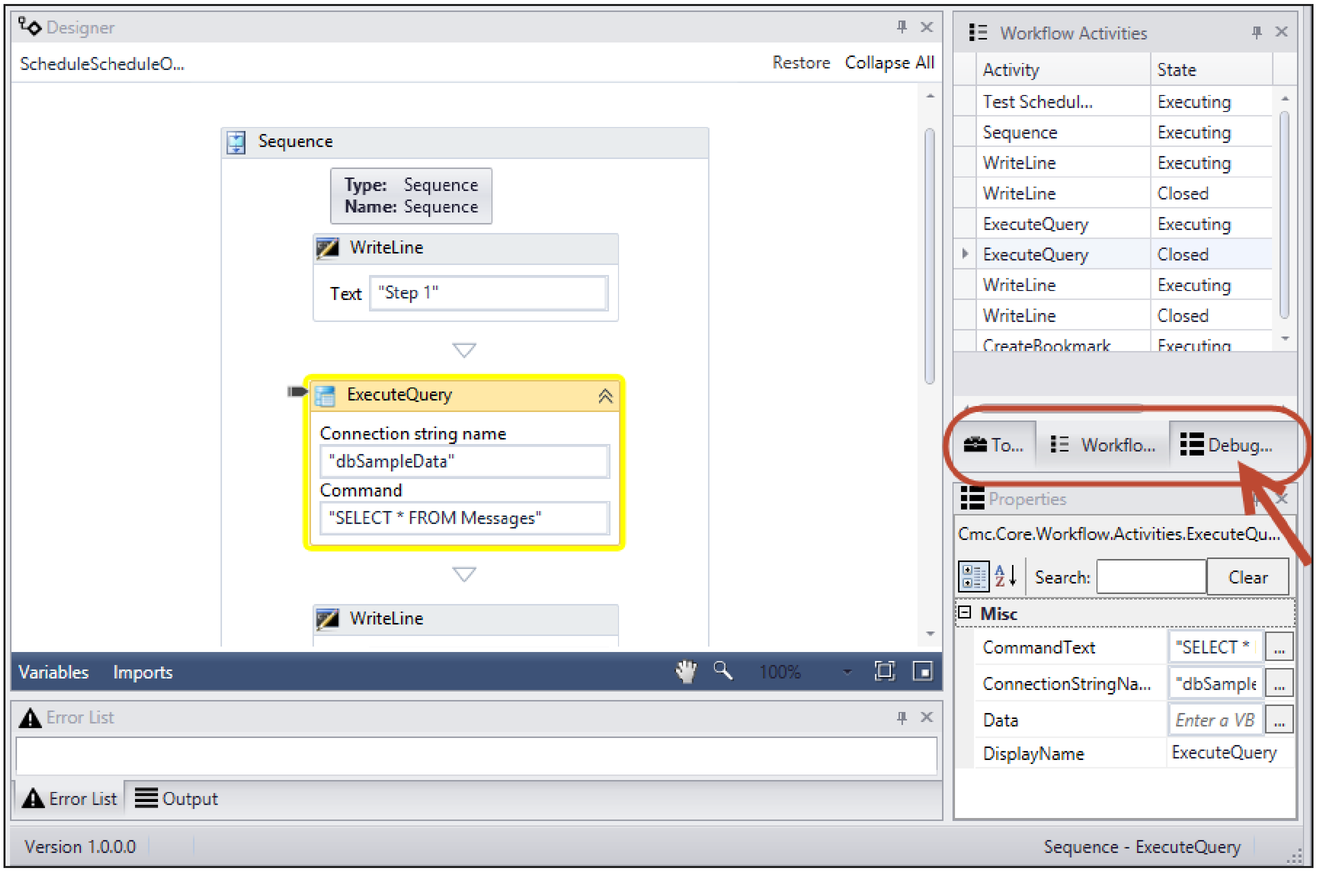
Command
"SELECT * FROM Messages"

Activity	State
Test Schedul...	Executing
Sequence	Executing
WriteLine	Executing
WriteLine	Closed
ExecuteQuery	Executing
ExecuteQuery	Closed
WriteLine	Executing
WriteLine	Closed
CreateBookmark	Executing

ImplementationVersi...	
Name	ScheduleSchedu

4. In the Workflow Activities pane, click on the **Activity** step that you want to examine. The selected Activity step is highlighted in the Designer pane.

5. Click on the **Debug Properties** tab.



Text "Step 1"

Connection string name
"dbSampleData"

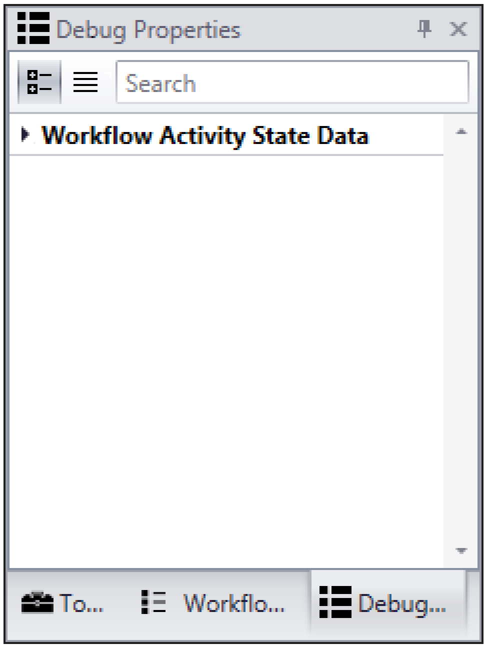
Command
"SELECT * FROM Messages"

Activity	State
Test Schedul...	Executing
Sequence	Executing
WriteLine	Executing
WriteLine	Closed
ExecuteQuery	Executing
ExecuteQuery	Closed
WriteLine	Executing
WriteLine	Closed
CreateBookmark	Executing

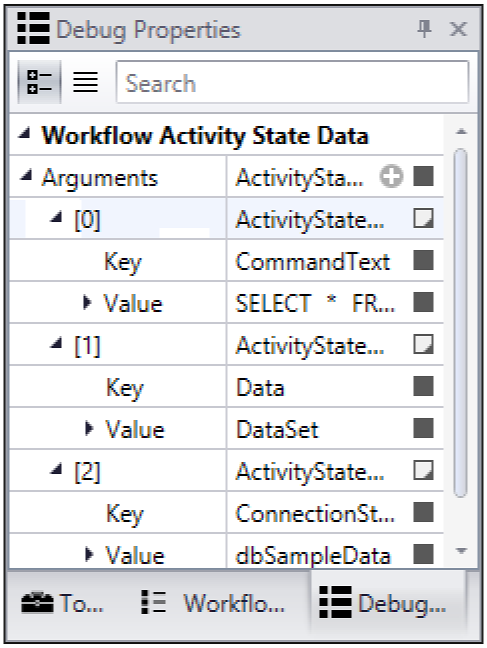
Cmc.Core.Workflow.Activities.ExecuteQu...

CommandText	"SELECT *
ConnectionStringNa...	"dbSample
Data	Enter a VB
DisplayName	ExecuteQuery

6. In the Debug Properties pane, click ▸ to the left of **Workflow Activity State Data** to inspect the details of the Arguments and Variables declared at the time of the execution of the Activity step selected in the Workflow Activities tab.



7. Click the ◪ icons to inspect the details of the Arguments and Variables declared at the time of the execution of the workflow Activity step selected in the Workflow Activities tab.



Workflow Activity State Data

Arguments	ActivitySta...	⊕ ■
[0]	ActivityState...	◪
Key	CommandText	■
Value	SELECT * FR...	■
[1]	ActivityState...	◪
Key	Data	■
Value	DataSet	■
[2]	ActivityState...	◪
Key	ConnectionSt...	■
Value	dbSampleData	■

Notes:

- Use the visual workflow tracking feature only when needed to avoid any performance impacts.
- Define an appropriate tracking profile to limit the number of tracking records that are emitted at runtime. For more information about tracking profiles, see http://msdn.microsoft.com/en-us/library/ee513989(v=vs.110).aspx.
- To clean up the Workflow Tracking database when it gets too large, refer to Resources > Workflow Tracking DB Cleanup Script.

New Workflows

About the New Object Model

Beginning with Workflow 2.2, a new object model supports Anthology Student version 17.1 and later. The new object model introduces new namespaces for Anthology Student modules.

Old Namespace	New Namespace

`Cmc.Nexus.Workflow.<modulename>`
Example:
`Cmc.Nexus.Workflow.Sis.Academics`
	`Cmc.Nexus.<modulename>.Workflow`
	Example:
	`Cmc.Nexus.Academics.Workflow`

The new services, namespaces, and entities are documented in the Anthology Student Object Library.

End-of-Life Announcement for Anthology Student Activities (V1)

With the release of Anthology Student 21.0 in October 2019, the EOL date for Anthology Student Activities (V1) is scheduled for October 2020 and the EOS date is scheduled for April 2021. For more information, see End-of-Life for Anthology Student Activities (V1).

New and Migrated Activities

The activities in the toolbox of Workflow Composer are sorted by namespace. Any new activities that have been developed since the introduction of the new object model are added to the corresponding namespaces in the toolbox.

Activities that were developed in the old object model and are required to support events raised out of Anthology Student were migrated to new namespaces.

Example:

The CreateStudentSportsService activity was migrated from `Cmc.Nexus.Workflow.Sis.StudentServices` to `Cmc.Nexus.StudentServices.Workflow`.

If you are creating a new workflow using this activity, use the activity from the new namespace `Cmc.Nexus.StudentServices.Workflow`.

For help about the migrated activity, refer to "CreateStudentSportsService **(V2)**" in the **New Workflows** help section.

Help about the older variant of the activity is found in "CreateStudentSportsService **(V1)**" in the **Legacy Workflows** help section.

The toolbox in Workflow Composer will provide both variants of the CreateStudentSportsService activity until all legacy workflows have been migrated.

The LookupServiceListItem, LookupAreaOfStudy, and LookupListItem activities were not migrated. The functionality of these activities is incorporated into the **LookupReferenceItem** activity in `Cmc.Nexus.Common.Workflow`. Use the LookupReferenceItem activity for any new or migrated workflows.

The LookupGroup activity in `Cmc.Nexus.Workflow` is migrated to LookupStudentGroup in `Cmc.Nexus.Common.Workflow`.

For detailed information about the entities and properties associated with new and migrated activities, refer to the Anthology Student Object Library instead of mapping tables provided in the *Legacy Workflows* help section.

Events

Events raised out of the standard interface for Anthology Student are supported only in the new object model.

Events raised out of the legacy interface for Anthology Student are supported in the legacy model (using legacy contracts, activities, and entity mapping tables). However, the legacy model will be phased out. Any new workflows for events raised out of the legacy interface for Anthology Student 17.1 and later should be migrated to use the new object model.

Contracts

The contracts that the legacy services/activities were developed against are not migrated. Instead, the contracts that the legacy services/activities use become part of the new object model/command model.

The legacy contracts will be supported for a designated length of time allowing for customers to adjust any applicable workflows to use the new entities and their corresponding contracts. The specific steps/process for how affected workflows are updated/modified will need to be determined.

If you are migrating from an older version of Anthology Student to a newer version, you may need to work with two instances of Workflow Composer where one instance uses the V1 and V2 packages of the older Anthology Student version and the second instance uses the V1 and V2 packages for the new Anthology Student version.

When all workflows are migrated to use the new activities, uninstall the old contracts. A new user from Anthology Student 17.1 forward should never install the old contracts/activities.

Converted Entities

In the new object model, the conversion of entity values is no longer required. The CVueIdToPersonIdActivity and PersonIdToCVueIdActivity are no longer needed, and the following conversion formulas no longer apply:

For Student:

- PersonId = (SyStudentId * 10) + 1

Other entities:

- SyStaffId + '2'
- SyAddressId + ‘3’
- PlEmployerContactId + ‘4’

- AmAgencyContactId + ‘5’
- SyOrganizationContactId + ‘6’
- AmOnlineApplicantId + ‘7’

For Student Group: GroupId = (SyGroupsId * 10) + 1

Note: In new and migrated workflows, the Campus (Id) property replaces the Business Unit (Id) property.

CampusNexus CRM Events

The following events are specific to CampusNexus CRM.

- Saving events are triggered just prior to data being saved.
- Saved events are triggered just after data is saved
- Deleting events are triggered just prior to data being deleted.
- Deleted events are triggered just after data is saved

These events apply to all operational objects except the Account object.

Note: In the current release, the Prospect object is renamed to Lead.

Cmc.NexusCrm.Contracts.dll

All operational and reference objects are wrapped in the assembly file Cmc.NexusCrm.Contracts.dll. Whenever new properties are created in CampusNexus CRM or an existing property definition (metadata) is changed, this assembly is regenerated. Workflows for CampusNexus CRM require the events and objects contained in the Cmc.NexusCrm.Contracts.dll to be available in Workflow Composer.

To regenerate the assembly after any metadata changes, perform the following steps:

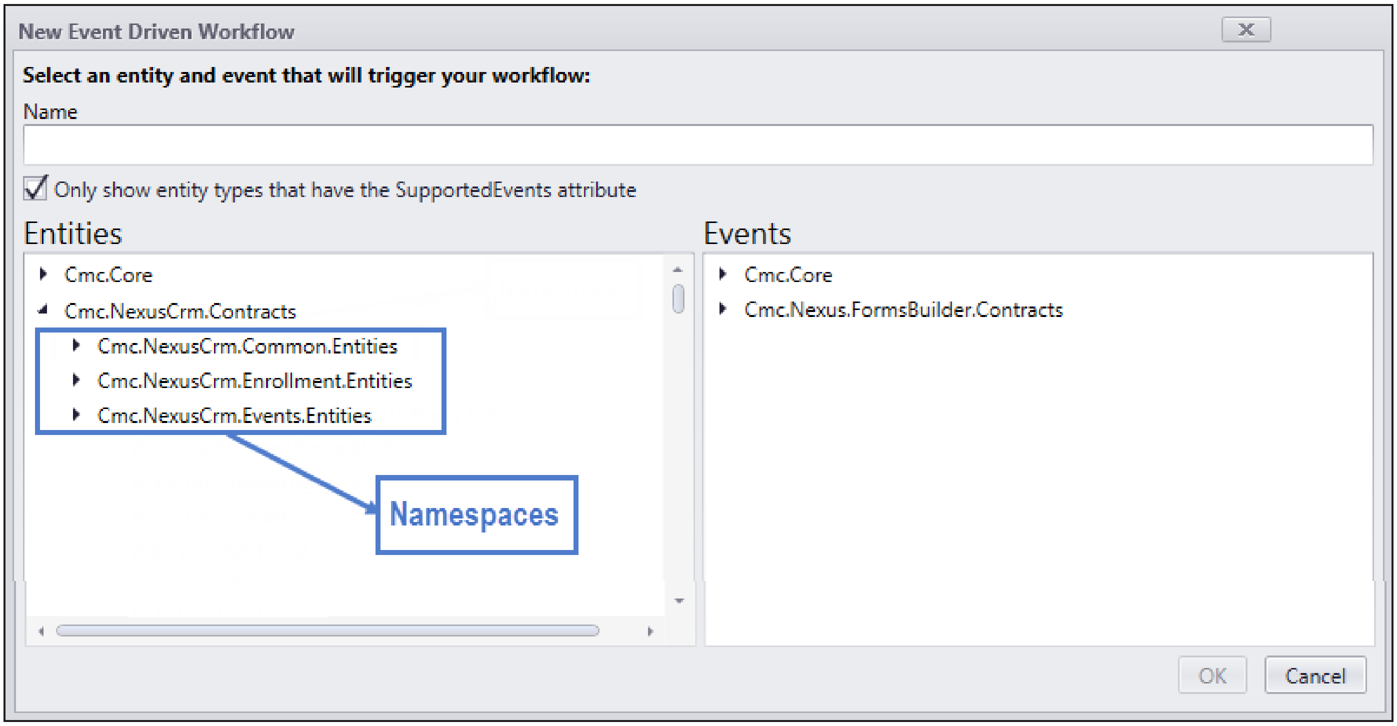
1. On the IIS Server of the Web Client for CampusNexus CRM, **restart** the **Cmc.Crm.Workspaces** application.
2. Navigate to the URL of the Web Client for CampusNexus CRM.
3. Copy the regenerated **Cmc.NexusCrm.Contracts.dll** from the \bin folder of the Web Client to the installation path of Workflow Composer.

CampusNexus CRM Namespaces

Entities of operational objects will be available under this contract in the following namespaces:

- Cmc.NexusCrm.Common.Entities
- Cmc.NexusCrm.Enrollment.Entities
- Cmc.NexusCrm.Events.Entities

The following figure is an example of a namespace:



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

Entities
Cmc.Core
Cmc.NexusCrm.Contracts
Cmc.NexusCrm.Common.Entities
Cmc.NexusCrm.Enrollment.Entities
Cmc.NexusCrm.Events.Entities
Namespaces
Events
Cmc.Core
Cmc.Nexus.FormsBuilder.Contracts
OK
Cancel

The following table indicates the list of objects supported in the above namespaces:

CRM Objects and Namespaces

ObjectName	Namespace	Events can occur in		
		Web Client	Portal	iServices
Account	Cmc.NexusCrm.Common.Entities	NA	NA	NA
Academic Progress	Cmc.NexusCrm.Enrollment.Entities	Y	Y	Y
Address	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Address Type	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Area of Interest	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Area of Study	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Contact	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Country	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Course History	Cmc.NexusCrm.Enrollment.Entities	Y	Y	Y
Custom Objects	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Document Status	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Document Status Type	Cmc.NexusCrm.Common.Entities	Y	Y	Y

ObjectName	Namespace	Events can occur in		
		Web Client	Portal	iServices
Education Level	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Enrollment	Cmc.NexusCrm.Enrollment.Entities	Y	Y	Y
Ethnic Group	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Event	Cmc.NexusCrm.Events.Entities	Y	Y	Y
Goal	Cmc.NexusCrm.Enrollment.Entities	Y	Y	Y
Lead	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Lead Source	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Lead Type	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Nationality	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Participant	Cmc.NexusCrm.Events.Entities	Y	Y	Y
Program	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Program Level	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Program Version	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Program Version Start Date	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Region	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Shift	Cmc.NexusCrm.Common.Entities	Y	Y	Y
State	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Term	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Test	Cmc.NexusCrm.Common.Entities	Y	Y	Y
Test Score	Cmc.NexusCrm.Enrollment.Entities	Y	Y	Y

Limitations:

- For the Event object, only the Get operation is supported.
- For the Participant object, only the Get and Update operations are supported.
- For all other objects, the Get, Create, and Update operations are supported.
- The Delete operation is not supported in all objects listed in the table.
- For external properties in all objects, only the Get activity is supported.

Deleting Events

Deleting events are triggered just before records are deleted. These events are used in scenarios where a user or an administrator needs to be notified prior to the deletion of a record.

For the handler written in .NET code to raise a validation, the second parameter, EventArgs, must be type casted to ValidationEventArgs.

Example for the Lead entity:

```
eventService.GetEvent<DeletingEvent>().RegisterHandler<Lead>(((lead, args) =>
   {
      var msg = (ValidationEventArgs) args;
      msg.ValidationMessages.Add("Not a valid ID");
   }
   ));
```

Anthology Student Database Events

In Workflow Composer 4.0 and later, the Service Module Host raises two new database events for Anthology Student:

- The **Database Row Deleted Event** occurs after a row in a database is deleted.
- The **Database Row Saved Event** occurs after a row in a database is updated or inserted.

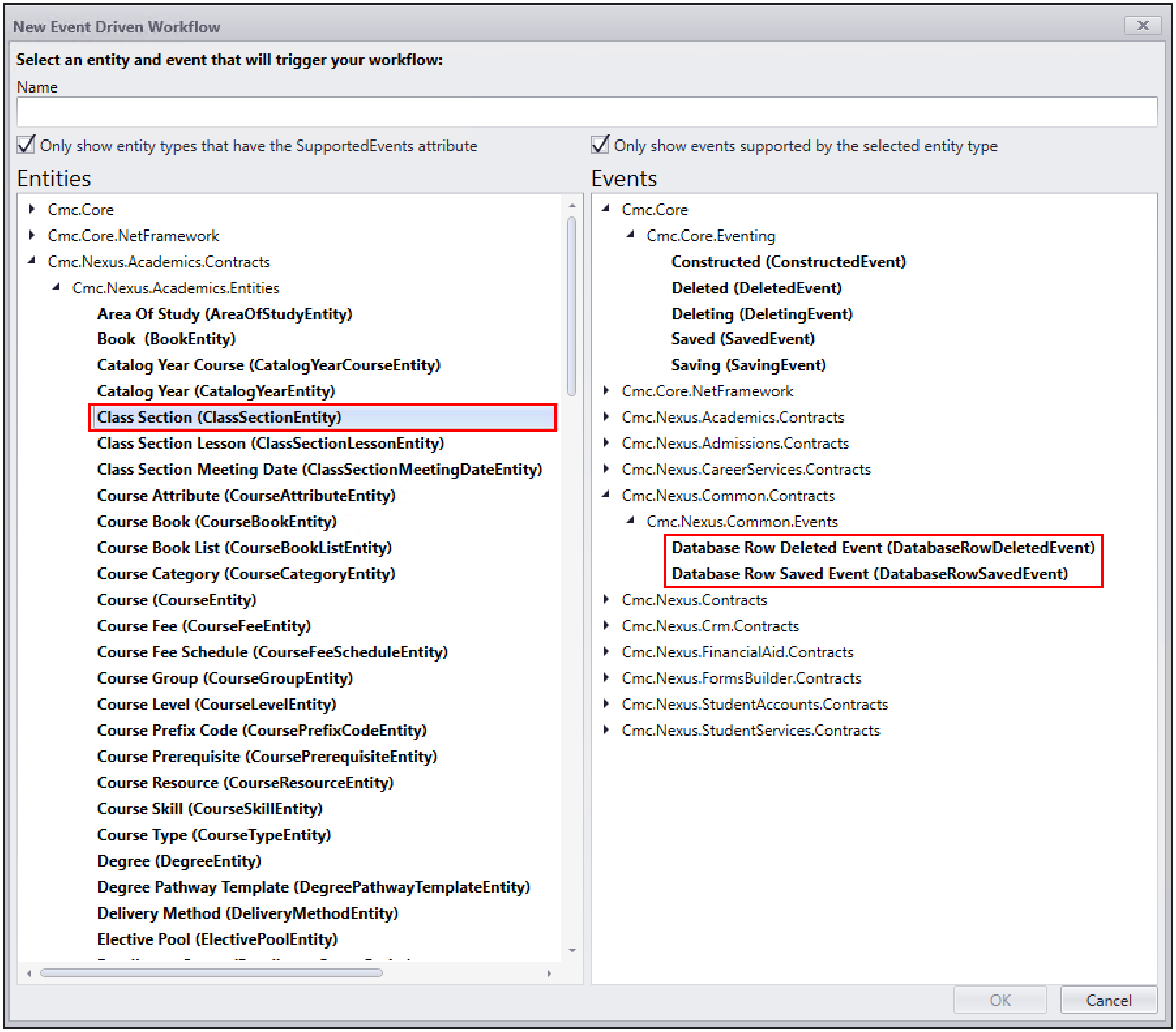
Previously, for some entities in the Anthology data model, the events raised from Anthology Student (standard interface) were insufficient to notify external systems of changes to a given entity. With the introduction of the new database events, additional data changes can be captured via raised events to support various integrations between Anthology Student and other systems.

Several Anthology Student contract entities are updated to serialize/deserialize the payload for the new database events. The first tables and entities that will support the new database event types are listed below. Other tables/entities will be added in the future.

Table	Entity	Event Added in Workflow 4.0
AdClassSched	Class Section Entity	Database Row Deleted Event Database Row Saved Event
AdConcentrationByEnrollment	Student Area Of Study Entity	Database Row Deleted Event Database Row Saved Event
SyStatChange	Student School Status History Entity	Database Row Saved Event

The new database events can be selected in Workflow Composer to create workflows for these entities.

Class Section Entity (Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities)



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

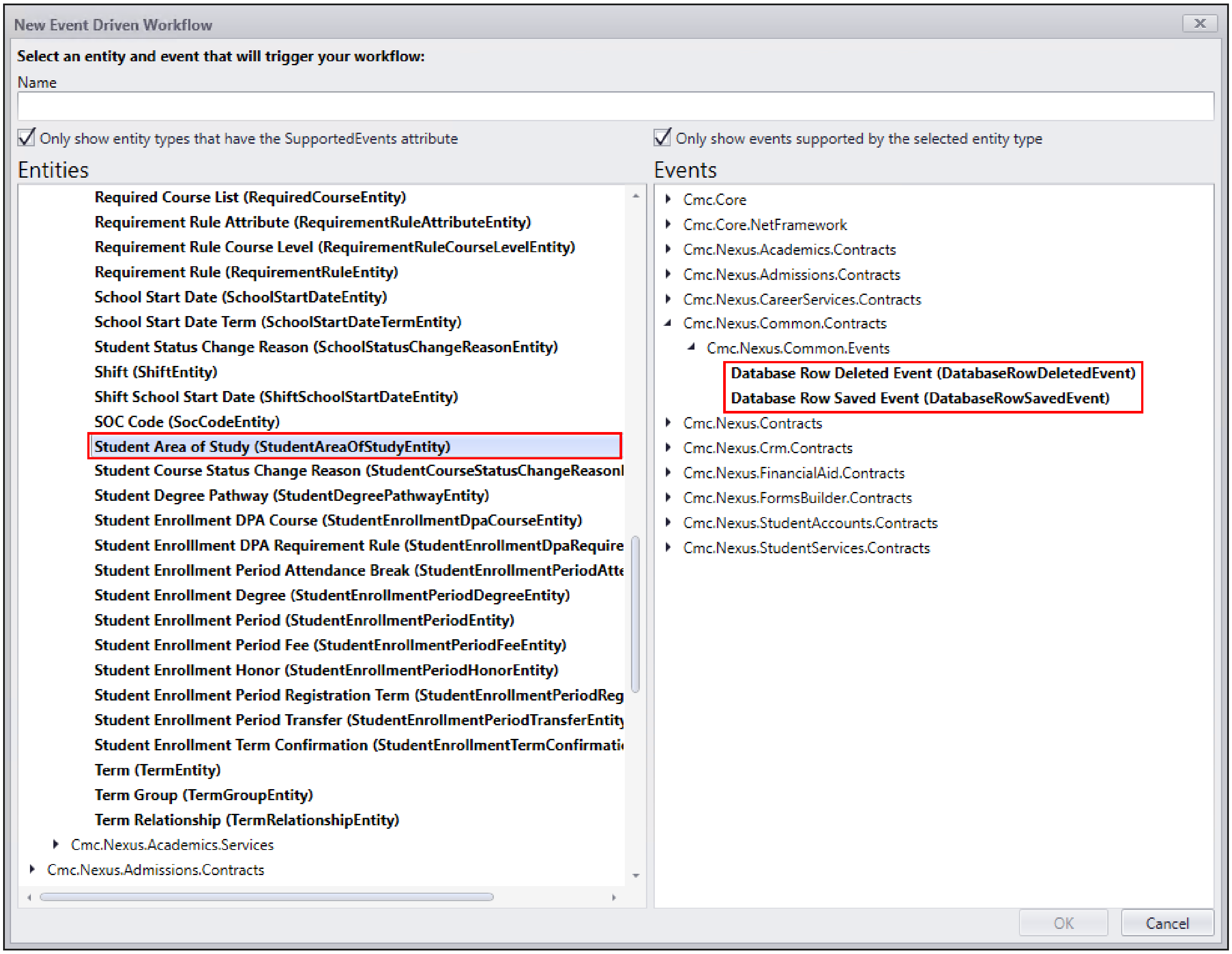
Entities

- Cmc.Core
- Cmc.Core.NetFramework
- Cmc.Nexus.Academics.Contracts
  - Cmc.Nexus.Academics.Entities
    - Area Of Study (AreaOfStudyEntity)
    - Book (BookEntity)
    - Catalog Year Course (CatalogYearCourseEntity)
    - Catalog Year (CatalogYearEntity)
    - Class Section (ClassSectionEntity)
    - Class Section Lesson (ClassSectionLessonEntity)
    - Class Section Meeting Date (ClassSectionMeetingDateEntity)
    - Course Attribute (CourseAttributeEntity)
    - Course Book (CourseBookEntity)
    - Course Book List (CourseBookListEntity)
    - Course Category (CourseCategoryEntity)
    - Course (CourseEntity)
    - Course Fee (CourseFeeEntity)
    - Course Fee Schedule (CourseFeeScheduleEntity)
    - Course Group (CourseGroupEntity)
    - Course Level (CourseLevelEntity)
    - Course Prefix Code (CoursePrefixCodeEntity)
    - Course Prerequisite (CoursePrerequisiteEntity)
    - Course Resource (CourseResourceEntity)
    - Course Skill (CourseSkillEntity)
    - Course Type (CourseTypeEntity)
    - Degree (DegreeEntity)
    - Degree Pathway Template (DegreePathwayTemplateEntity)
    - Delivery Method (DeliveryMethodEntity)
    - Elective Pool (ElectivePoolEntity)

Events

- Cmc.Core
  - Cmc.Core.Eventing
    - Constructed (ConstructedEvent)
    - Deleted (DeletedEvent)
    - Deleting (DeletingEvent)
    - Saved (SavedEvent)
    - Saving (SavingEvent)
- Cmc.Core.NetFramework
- Cmc.Nexus.Academics.Contracts
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.CareerServices.Contracts
- Cmc.Nexus.Common.Contracts
  - Cmc.Nexus.Common.Events
    - Database Row Deleted Event (DatabaseRowDeletedEvent)
    - Database Row Saved Event (DatabaseRowSavedEvent)
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.FormsBuilder.Contracts
- Cmc.Nexus.StudentAccounts.Contracts
- Cmc.Nexus.StudentServices.Contracts

Student Area Of Study Entity (Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities)



Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

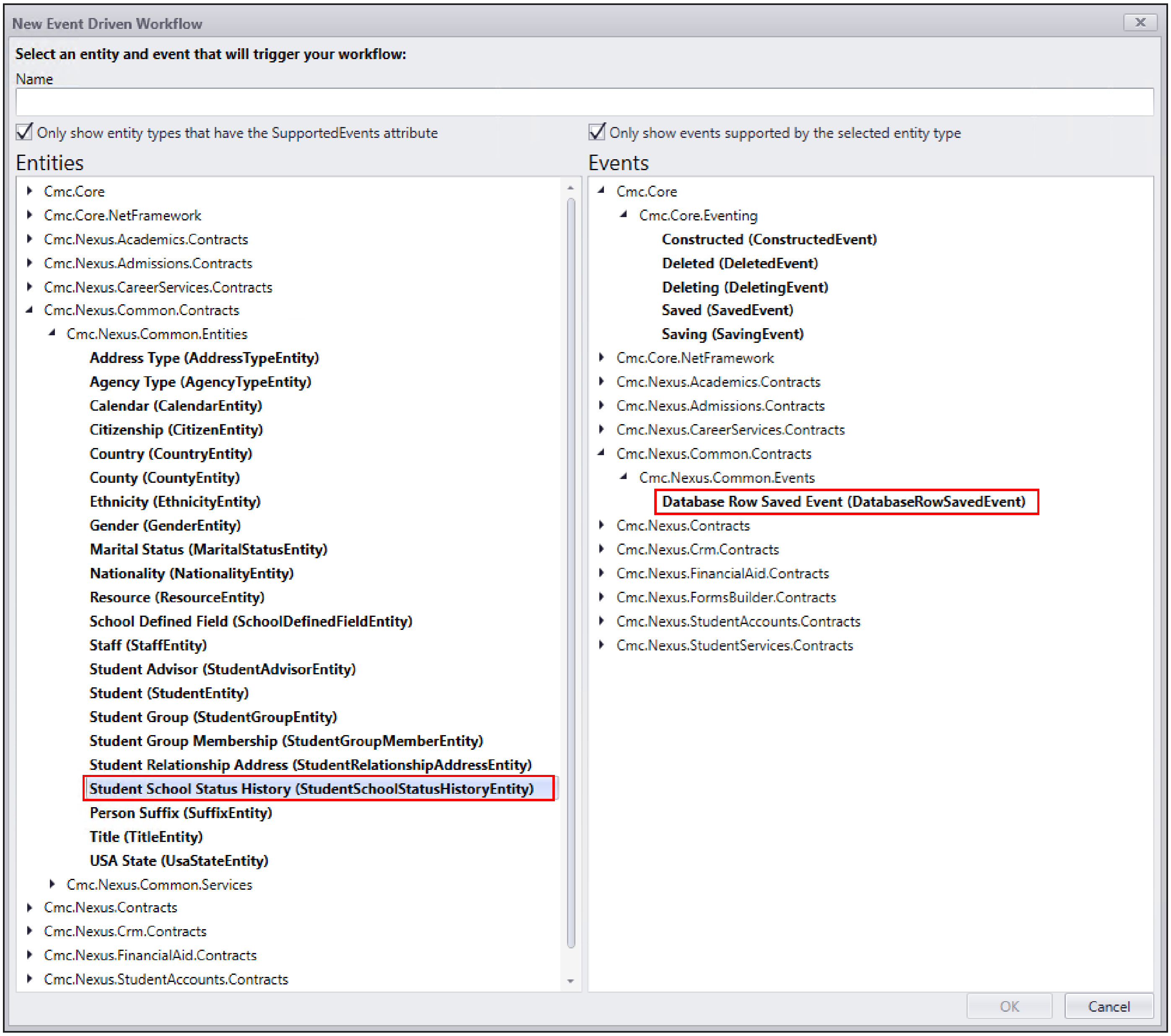
Entities

- Required Course List (RequiredCourseEntity)
- Requirement Rule Attribute (RequirementRuleAttributeEntity)
- Requirement Rule Course Level (RequirementRuleCourseLevelEntity)
- Requirement Rule (RequirementRuleEntity)
- School Start Date (SchoolStartDateEntity)
- School Start Date Term (SchoolStartDateTermEntity)
- Student Status Change Reason (SchoolStatusChangeReasonEntity)
- Shift (ShiftEntity)
- Shift School Start Date (ShiftSchoolStartDateEntity)
- SOC Code (SocCodeEntity)
- Student Area of Study (StudentAreaOfStudyEntity)
- Student Course Status Change Reason (StudentCourseStatusChangeReasonI
- Student Degree Pathway (StudentDegreePathwayEntity)
- Student Enrollment DPA Course (StudentEnrollmentDpaCourseEntity)
- Student Enrolllment DPA Requirement Rule (StudentEnrollmentDpaRequire
- Student Enrollment Period Attendance Break (StudentEnrollmentPeriodAtte
- Student Enrollment Degree (StudentEnrollmentPeriodDegreeEntity)
- Student Enrollment Period (StudentEnrollmentPeriodEntity)
- Student Enrollment Period Fee (StudentEnrollmentPeriodFeeEntity)
- Student Enrollment Honor (StudentEnrollmentPeriodHonorEntity)
- Student Enrollment Period Registration Term (StudentEnrollmentPeriodReg
- Student Enrollment Period Transfer (StudentEnrollmentPeriodTransferEntity
- Student Enrollment Term Confirmation (StudentEnrollmentTermConfirmati
- Term (TermEntity)
- Term Group (TermGroupEntity)
- Term Relationship (TermRelationshipEntity)
- Cmc.Nexus.Academics.Services
- Cmc.Nexus.Admissions.Contracts

Events

- Cmc.Core
- Cmc.Core.NetFramework
- Cmc.Nexus.Academics.Contracts
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.CareerServices.Contracts
- Cmc.Nexus.Common.Contracts
  - Cmc.Nexus.Common.Events
    - Database Row Deleted Event (DatabaseRowDeletedEvent)
    - Database Row Saved Event (DatabaseRowSavedEvent)
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.FormsBuilder.Contracts
- Cmc.Nexus.StudentAccounts.Contracts
- Cmc.Nexus.StudentServices.Contracts

Student School Status History Entity (Cmc.Nexus.Common.Contracts > Cmc.Nexus.Common.Entities)



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

Entities

- Cmc.Core
- Cmc.Core.NetFramework
- Cmc.Nexus.Academics.Contracts
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.CareerServices.Contracts
- Cmc.Nexus.Common.Contracts
  - Cmc.Nexus.Common.Entities
    - **Address Type (AddressTypeEntity)**
    - **Agency Type (AgencyTypeEntity)**
    - **Calendar (CalendarEntity)**
    - **Citizenship (CitizenEntity)**
    - **Country (CountryEntity)**
    - **County (CountyEntity)**
    - **Ethnicity (EthnicityEntity)**
    - **Gender (GenderEntity)**
    - **Marital Status (MaritalStatusEntity)**
    - **Nationality (NationalityEntity)**
    - **Resource (ResourceEntity)**
    - **School Defined Field (SchoolDefinedFieldEntity)**
    - **Staff (StaffEntity)**
    - **Student Advisor (StudentAdvisorEntity)**
    - **Student (StudentEntity)**
    - **Student Group (StudentGroupEntity)**
    - **Student Group Membership (StudentGroupMemberEntity)**
    - **Student Relationship Address (StudentRelationshipAddressEntity)**
    - **Student School Status History (StudentSchoolStatusHistoryEntity)**
    - **Person Suffix (SuffixEntity)**
    - **Title (TitleEntity)**
    - **USA State (UsaStateEntity)**
  - Cmc.Nexus.Common.Services
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.StudentAccounts.Contracts

Events

- Cmc.Core
  - Cmc.Core.Eventing
    - **Constructed (ConstructedEvent)**
    - **Deleted (DeletedEvent)**
    - **Deleting (DeletingEvent)**
    - **Saved (SavedEvent)**
    - **Saving (SavingEvent)**
- Cmc.Core.NetFramework
- Cmc.Nexus.Academics.Contracts
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.CareerServices.Contracts
- Cmc.Nexus.Common.Contracts
  - Cmc.Nexus.Common.Events
    - **Database Row Saved Event (DatabaseRowSavedEvent)**
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.FormsBuilder.Contracts
- Cmc.Nexus.StudentAccounts.Contracts
- Cmc.Nexus.StudentServices.Contracts

Event Details

Multiple Triggers

Database Row Saved/Deleted events trigger workflows multiple times due to other processes, triggers, and stored procedures that affect the database record. If a user had included an email notification in these workflows, multiple notifications would be received for each Database Row Saved/Deleted event.

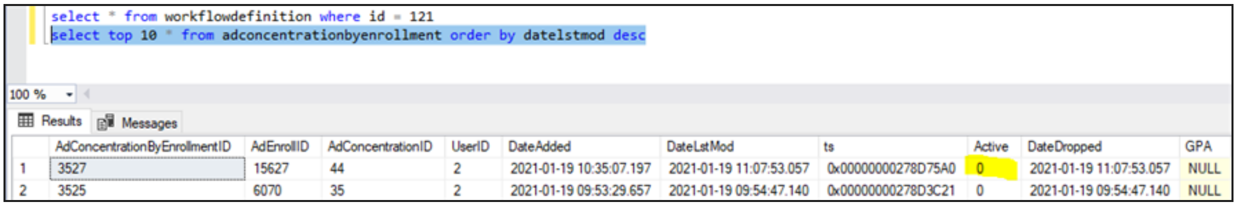
Database Row Saved Events:

- On the Class Section entity, the workflow is triggered 8 times.
- On the Student Area Of Study entity, the workflow is triggered 8 times.

- On the Student School Status History entity, the workflow is triggered 4 times.

Database Row Deleted Events:

- On the Class Section entity, the workflow is triggered 4 times.
- On the Student Area Of Study entity, the workflow is triggered 0 times. The event fires if the record is deleted manually in the database. The workflow just sets the record to inactive (Active=0) as shown below.



```
select * from workflowdefinition where id = 121
select top 10 * from adconcentrationbyenrollment order by datelstmod desc
```

	AdConcentrationByEnrollmentID	AdEnrollID	AdConcentrationID	UserID	DateAdded	DateLstMod	ts	Active	DateDropped	GPA
1	3527	15627	44	2	2021-01-19 10:35:07.197	2021-01-19 11:07:53.057	0x00000000278D75A0	0	2021-01-19 11:07:53.057	NULL
2	3525	6070	35	2	2021-01-19 09:53:29.657	2021-01-19 09:54:47.140	0x00000000278D3C21	0	2021-01-19 09:54:47.140	NULL

Logging

Workflow logs for database row events will not include values for date created and date modified. The DateTime values will only appear in the database after the event is fired. The workflow just logs the event object.

```
2021-01-14 14:40:18.6921   67 Error                         Cmc.Core.Workflow.Activities.LogLine
Student AOS SAVED
2021-01-14 14:40:18.6921   67 Error                         Cmc.Core.Work-
flow.Activities.LogObject {
  "IsExcludedCrmIntegration": false,
  "Id": 3524,
  "AreaOfStudyId": 43,
  "CreatedByUserId": 2,
"CreatedDateTime": "0001-01-01T00:00:00",
  "DropByUserId": 0,
  "DropDate": null,
  "Gpa": null,
  "IsActive": true,
  "LastModifiedDateTime": "0001-01-01T00:00:00",
  "LastModifiedUserId": 2,
  "ProgramVersionAreaOfStudyId": 428,
  "RowVersion": null,
"StudentAreaOfStudyParentId": 0,
  "StudentEnrollmentPeriodId": 15627,
  "OriginalState": "H4sIAAAAAAAEALWUTWuEMBBA/8rivcb-
t3sQV7HYP0l0UbJde-
gxndQD5kMtZNf31dC6UXeyjklJDJvJcJk2QFIvdV9wL+wtUIVecIpem58a9+gM1NK+P20ZVoSBlz7RU0d7GWLVpnO4p-
bq9ljkuzYNmENoORKfnKS1rAF7KJvQip/GNM0xdMut-
tjfE7f-
s/XxqFuyDNI64aSHKs5XjLIG8FBm7j9myZSNTI9U+Ih-
whYvkSWs1d-
hxYIvOoaGoUPwj/MfALx5N8cYBDDM9o-
hIL50RUvyA4Kgj7dWjQLEAXVpCHp-
ceiiA6sQdna2QnQQR7KpqtHMF+gLo5ipCd9adDIZ+aWqO80JI2dGgVUrPs3p+9Fb818X+/n7yL2iiMkifBAAA",
  "SecureState":
"H4sIAAAAAAAEAD3MQQrCMBAAwK9I7mZTey-
sqeFb-
poSJel5C0C01Ss-
isxvl6s4ANm9qec-
sfb+7Ood56-
frPUumOGKst7q4zSvMkQ9qElk6ALaTC8g6kM2JkxdtU4CdMS00BgaXCWd6o1CKsMas-
fkNH/6OUokurUx6/sIHH9TKs7ZYiC0brFBw/juQnSpcAAAA=",
  "ExtendedProperties": [],
  "EntityState": 0
}
```

Cmc.Nexus.Models

The following table shows entity mapping for the LookupReferenceItem activity (reference item query model).

Reference Item Type	Entity	Database Table
Account Statuses	Cmc.Nexus.Models.StudentAccounts.AccountStatus	SaAcctStatus
Address Types	Cmc.Nexus.Models.Common.AddressType	SyAddrType
Agencies	Cmc.Nexus.Models.Common.Agency	AmAgency
Applicant Types	Cmc.Nexus.Models.Admissions.ApplicantType	AmApplicantType
Area of Study Types	Cmc.Nexus.Models.Academics.AreaOfStudyType	AdConcentrationType
Areas Of Study	Cmc.Nexus.Models.Academics.AreaOfStudy	AdConcentration
Athletic Status	Cmc.Nexus.Models.StudentServices.AthleticStatus	SsAthleticStatus
Billing Methods	Cmc.Nexus.Models.StudentAccounts.BillingMethod	SaBillingMethod
Books for Course	Cmc.Nexus.Models.Academics.Books	BsItem
Campuses	Cmc.Nexus.Models.Common.Campus	SyCampus
Catalog Years	Cmc.Nexus.Models.Academics.CatalogYear	AdCatalogYear
CitizenCodes	Cmc.Nexus.Models.Common.Citizen	AmCitizen
Counties	Cmc.Nexus.Models.Common.County	SyCounty
Countries	Cmc.Nexus.Models.Common.Country	SyCountry
Customer Banks	Cmc.Nexus.Models.StudentAccounts.Bank	SaBank
Disability Statuses	Cmc.Nexus.Models.StudentServices.DisabilityStatus	SsDisabilityStatus
Document Statuses	Cmc.Nexus.Models.Crm.DocumentStatus	CmDocStatus
Document Types	Cmc.Nexus.Models.Crm.DocumentType	CmDocType
Employment Statuses	Cmc.Nexus.Models.CareerServices.EmploymentStatus	PlEmpStatus
Ethnicities	Cmc.Nexus.Models.Common.Ethnicity	AmRace
Fund Sources	Cmc.Nexus.Models.FinancialAid.FundSource	FaFundSource
Genders	Cmc.Nexus.Models.Common.Gender	AmSex
Grade Levels	Cmc.Nexus.Models.Academics.GradeLevel	AdGradeLevel
Grade Scales	Cmc.Nexus.Models.Academics.GradeScale	AdGradeScale

Reference Item Type	Entity	Database Table
Lead Source Categories	Cmc.Nexus.Models.Admissions.LeadCategory	AmLeadCat
Lead Sources	Cmc.Nexus.Models.Admissions.LeadSource	AmLeadSrc
Lead Types	Cmc.Nexus.Models.Admissions.LeadType	AmLeadType
Marital Statuses	Cmc.Nexus.Models.Common.MaritalStatus	AmMarital
Modules	Cmc.Nexus.Models.Common.Module	SyModule
Nationalities	Cmc.Nexus.Models.Common.Nationality	AmNationality
Previous Education Codes	Cmc.Nexus.Models.Admissions.PreviousEducation	AmPrevEduc
Programs	Cmc.Nexus.Models.Academics.Program	AdProgram
SAP Statuses	Cmc.Nexus.Models.Academics.SapStatus	AdSapStatus
School Start Dates	Cmc.Nexus.Models.Academics.SchoolStartDate	AdStartDate
School Status Change Reasons	Cmc.Nexus.Models.Academics.SchoolStatusChangeReason	AdReason
Service Types Categories	Cmc.Nexus.Models.StudentServices.ServiceTypeCategory	SsServiceCategory
Shifts	Cmc.Nexus.Models.Academics.Shift	AdShift
Sports	Cmc.Nexus.Models.StudentServices.Sport	SsSports
Staff	Cmc.Nexus.Models.Common.Staff	SyStaff
Staff Groups	Cmc.Nexus.Models.Common.StaffGroup	SyStaffGroup
Subsidiary Account Types	Cmc.Nexus.Models.StudentAccounts.SubsidiaryAccountType	SaSubsidiary
System School Statuses	Cmc.Nexus.Models.Common.SystemSchoolStatus	SyStatus
Task Results	Cmc.Nexus.Models.Crm.TaskResult	CmEventResult
Task Statuses	Cmc.Nexus.Models.Crm.TaskStatus	CmEventStatus
Task Templates	Cmc.Nexus.Models.Crm.TaskTemplate	CmTemplate
Task Types	Cmc.Nexus.Models.Crm.TaskType	CmEventType
Transaction Codes	Cmc.Nexus.Models.StudentAccounts.BillingTransactionCode	SaBillCode

CMC Activities

Workflow Designer is built using the Windows Workflow Foundation (WF) in the .NET Framework. It contains Microsoft's built-in (generic) workflow activities and activities created specifically for Anthology Inc. products (CMC Activities).

The workflow activities designed for Anthology are grouped by namespaces. The activities include lookup functions that return values that can in turn be used within other activities in the workflow, activities related to specific products such as CampusNexus CRM and Anthology Student, and common activities such as creating validation messages or sending email. CMC activities are used in conjunction with Generic Activities.

Properties for activities are generally defined using expressions in VB .NET code or variables. Some fields have drop-down lists and helpers that enable you to select properties.

Filter Option for Assemblies

Many workflow activities require the user to browse for and select a .NET type from the Anthology domain model. To improve the performance of the "Browse for Types..." action, the list of assemblies from which a user can select types is filtered down to just those that need to be used in Workflow Composer.

The "FilterUsableAssemblies" setting in the WorkflowComposer.exe.config file controls the filtering of assemblies. The default setting for the "FilterUsableAssemblies" value is `True`.

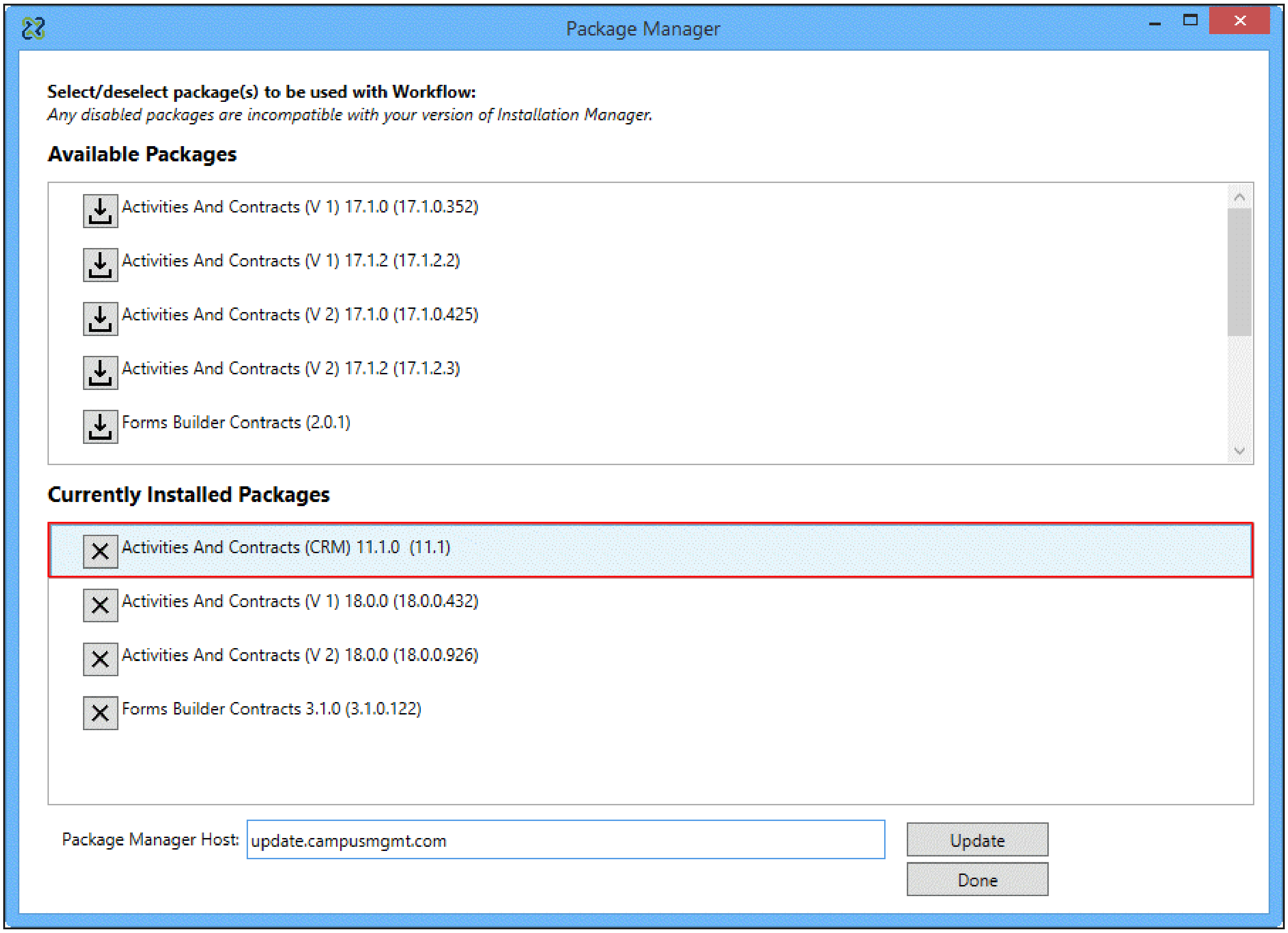
```
<setting name="FilterUsableAssemblies" serializeAs="String">
<value>True</value>
</setting>
```

If you need a namespace or type which is being filtered out, set the "FilterUsableAssemblies" value to **`False`** and restart Workflow Composer.

Note: When the filter option is disabled, the performance of the "Browse for Types..." action will be noticeably slower. To compensate for the performance loss, see Improve Search Performance on "Browse for Types...".

Activities for CampusNexus CRM

The activities in the Cmc.NexusCrm.Common.Workflow namespace are available when the Activities and Contracts (CRM) package is installed using the Package Manager.



Select/deselect package(s) to be used with Workflow:
Any disabled packages are incompatible with your version of Installation Manager.

Available Packages

- Activities And Contracts (V 1) 17.1.0 (17.1.0.352)
- Activities And Contracts (V 1) 17.1.2 (17.1.2.2)
- Activities And Contracts (V 2) 17.1.0 (17.1.0.425)
- Activities And Contracts (V 2) 17.1.2 (17.1.2.3)
- Forms Builder Contracts (2.0.1)

Currently Installed Packages

- Activities And Contracts (CRM) 11.1.0 (11.1)
- Activities And Contracts (V 1) 18.0.0 (18.0.0.432)
- Activities And Contracts (V 2) 18.0.0 (18.0.0.926)
- Forms Builder Contracts 3.1.0 (3.1.0.122)

Package Manager Host: update.campusmgmt.com

Prerequisite for CampusNexus CRM Workflows

The generated CampusNexus CRM contracts need to be copied to Workflow Composer for building and creating workflows. As a best practice, when CampusNexus CRM metadata is changed, the generated contracts assembly file (**Cmc.NexusCrm.Contracts.dll**) must be copied from the \bin folder of the Web Client for CampusNexus CRM to the installation path of Workflow Composer.

If an existing workflow includes a property that is not available in the current generated contracts, the administrator needs to manually edit the workflow and remove the property.

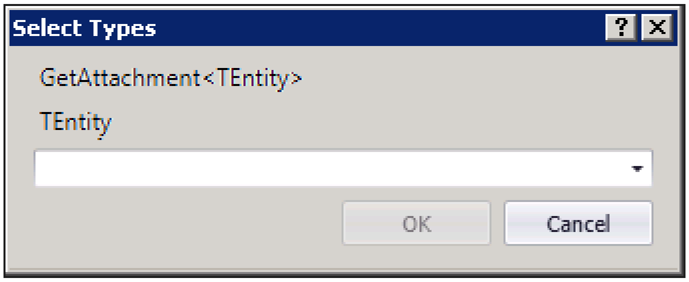
Cmc.NexusCrm.Common.Workflow

Workflow activities specific to CampusNexus CRM are grouped under the Cmc.NexusCrm.Common.Workflow namespace. The activities include get functions that enable you to retrieve attachments and related entities, and a lookup function that returns contact ID values that are consumed in Forms Builder.

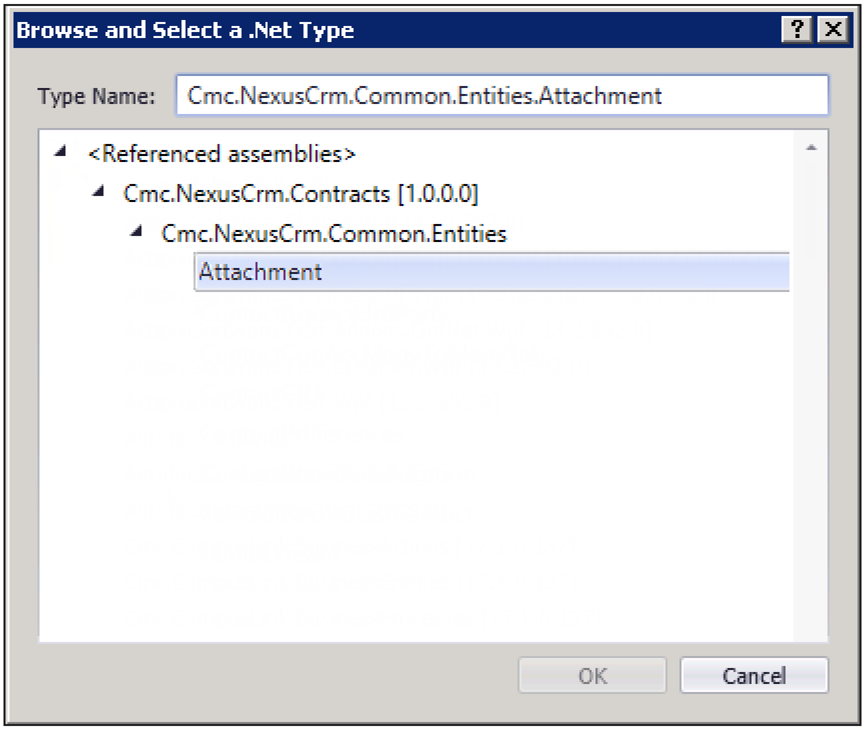
GetAttachment<>

> ⚠ The GetRelatedEntity<> activity must be included in the workflow before the GetAttachment<> activity, and the GetEntity<> activity must precede the GetRelatedEntity<> activity. Attachments in the tab retrieved from the GetRelatedEntity<> activity are retrieved in the GetAttachment<> activity.

The GetAttachment<> activity retrieves attachments from the Id of the tab that is retrieved in the GetRelatedEntity<> activity.



When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.


Browse and Select a .Net Type
?
<Referenced assemblies>
Cmc.NexusCrm.Contracts [1.0.0.0]
Cmc.NexusCrm.Common.Entities
Attachment
OK
Cancel

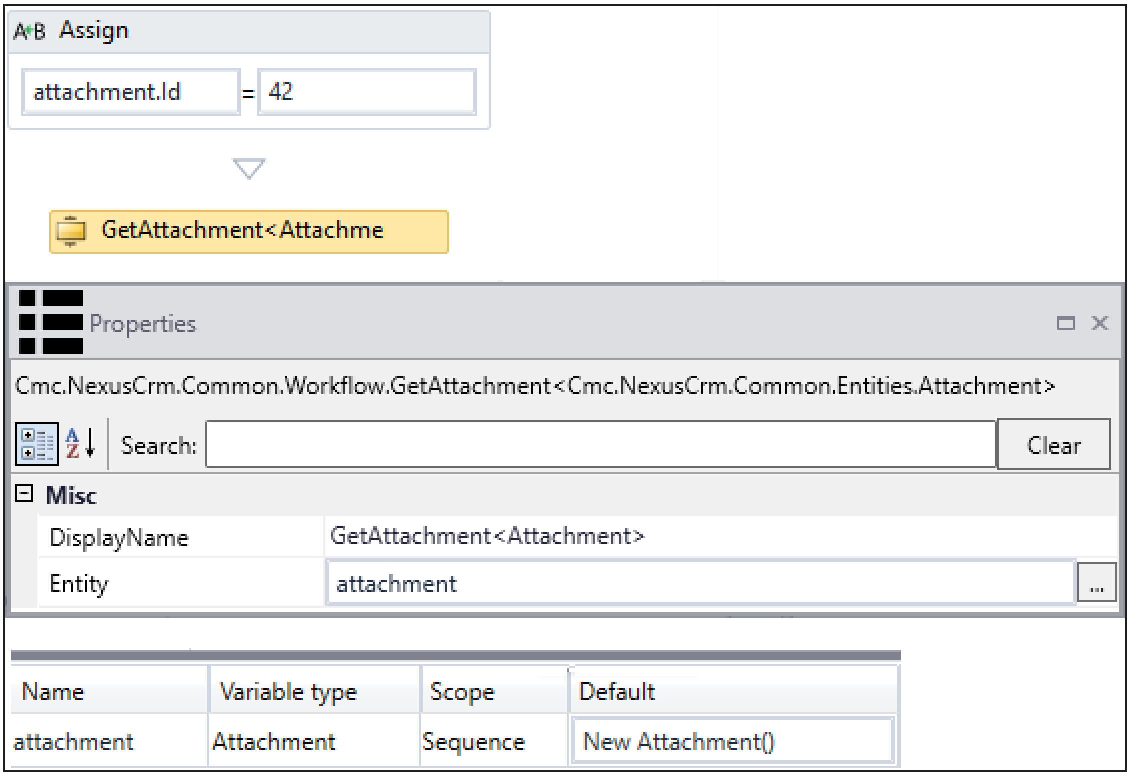
Type Name: Cmc.NexusCrm.Common.Entities.Attachment

After you have selected an entity, the name of the entity is inserted in the DisplayName field, e.g., GetAttachment <Attachment>. Proceed to specify the entity to be retrieved.

Scenario

To retrieve the attachment in an encrypted format, you are required to create a sequence of three activities:

- GetEntity<> — this activity retrieves the instance of the object record.
- GetRelatedEntity<> — this activity retrieves the ID of the tab from which you want to retrieve the attachment.
- GetAttachment<> — the ID of the tab serves as an input parameter. This activity then retrieves the attachment in an encrypted form.



attachment.Id = 42

GetAttachment<Attachme

Cmc.NexusCrm.Common.Workflow.GetAttachment<Cmc.NexusCrm.Common.Entities.Attachment>

DisplayName GetAttachment<Attachment>

Entity attachment

Name	Variable type	Scope	Default
attachment	Attachment	Sequence	New Attachment()

Properties

GetAttachment<> Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Entity	`InOutArgument<Entity>`	Yes	Specify the entity identifier using a VB expression or variable.

GetRelatedEntity<>

⚠ The GetEntity<> activity must precede the GetRelatedEntity<> activity.

The GetRelatedEntity<> activity retrieves logical identifiers of records in the specified recordlist tab of the object record that was retrieved in the GetEntity<> activity.

For an object, the GetRelatedEntity<> activity retrieves the following details:

From a RecordList tab:

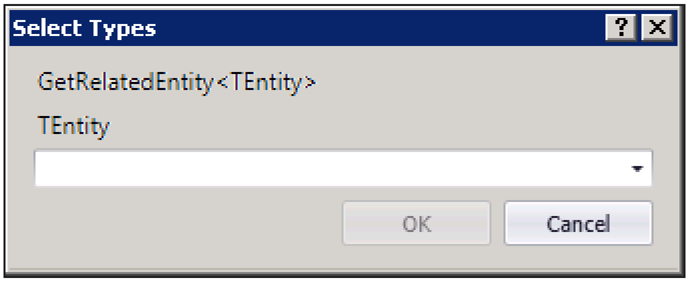
1. Instance Id of the Object
2. Row Id of the RecordList Property Value
3. RecordList Property Values

From a tab of a Many-To-Many relationship without relationship properties:

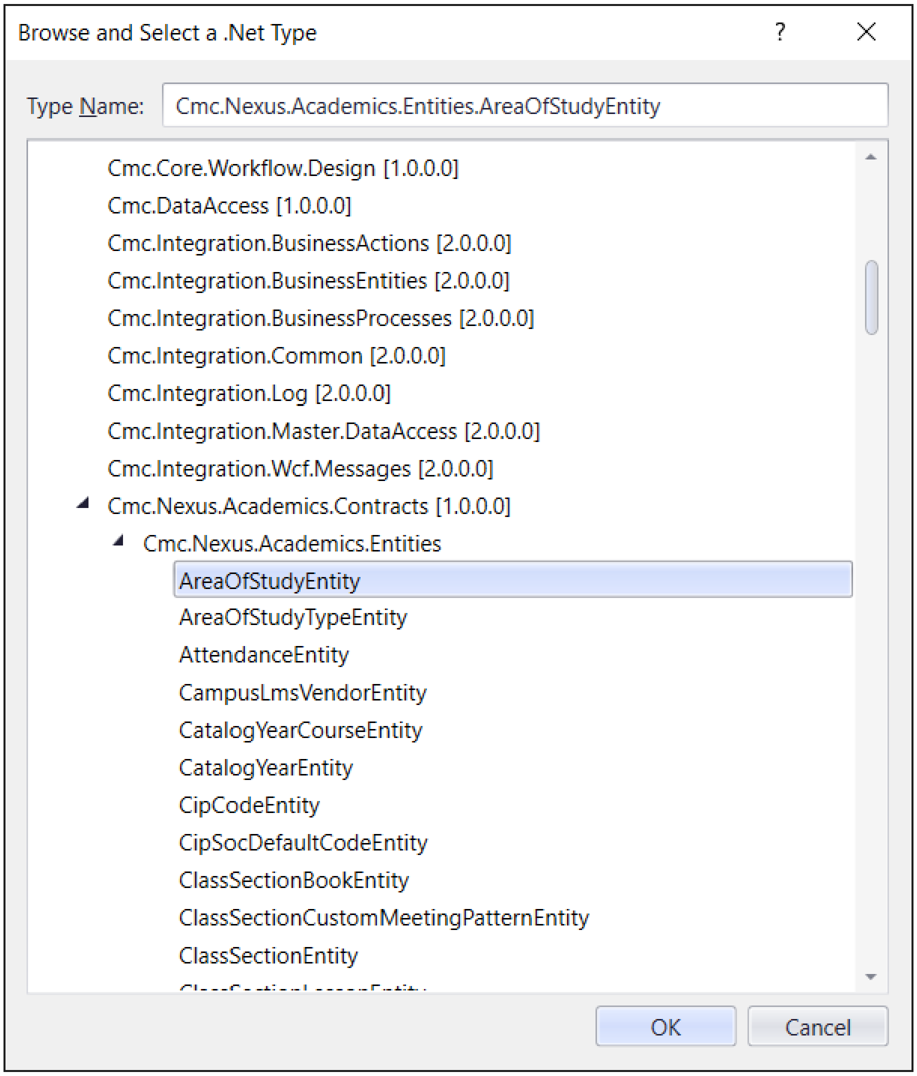
1. BaseObject Instance Id
2. Related Object Instance Id

From a tab of a Many-To-Many relationship with relationship properties:

1. BaseObject Instance Id
2. Related Object Instance Id
3. Associated relationship property value



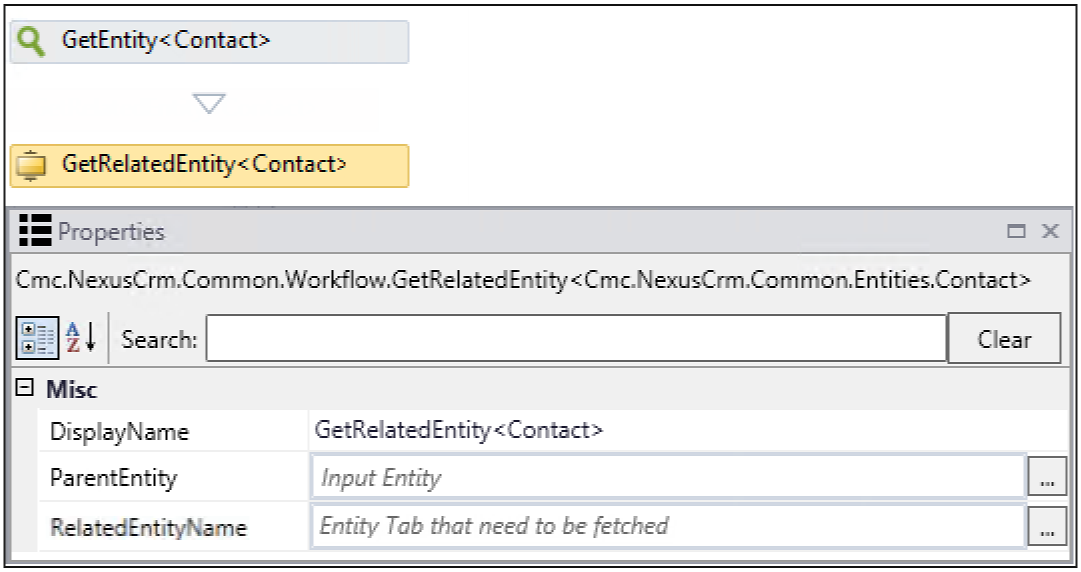
When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.



Type Name: Cmc.Nexus.Academics.Entities.AreaOfStudyEntity

- Cmc.Core.Workflow.Design [1.0.0.0]
- Cmc.DataAccess [1.0.0.0]
- Cmc.Integration.BusinessActions [2.0.0.0]
- Cmc.Integration.BusinessEntities [2.0.0.0]
- Cmc.Integration.BusinessProcesses [2.0.0.0]
- Cmc.Integration.Common [2.0.0.0]
- Cmc.Integration.Log [2.0.0.0]
- Cmc.Integration.Master.DataAccess [2.0.0.0]
- Cmc.Integration.Wcf.Messages [2.0.0.0]
- Cmc.Nexus.Academics.Contracts [1.0.0.0]
  - Cmc.Nexus.Academics.Entities
    - AreaOfStudyEntity
    - AreaOfStudyTypeEntity
    - AttendanceEntity
    - CampusLmsVendorEntity
    - CatalogYearCourseEntity
    - CatalogYearEntity
    - CipCodeEntity
    - CipSocDefaultCodeEntity
    - ClassSectionBookEntity
    - ClassSectionCustomMeetingPatternEntity
    - ClassSectionEntity

After you have selected an entity, the name of the entity is inserted in the DisplayName field, e.g., GetRelatedEntity <Contact>. Proceed to specify the entity to be retrieved, and the related entity name details.



Cmc.NexusCrm.Common.Workflow.GetRelatedEntity<Cmc.NexusCrm.Common.Entities.Contact>

DisplayName	GetRelatedEntity<Contact>
ParentEntity	*Input Entity*
RelatedEntityName	*Entity Tab that need to be fetched*

Properties

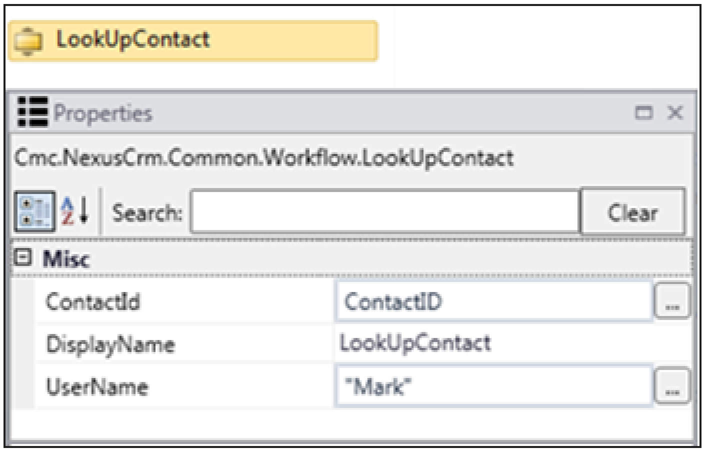
GetRelatedEntity<> Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Entity	`InOutArgument<Entity>`	Yes	Specify the entity identifier using a VB expression or variable.
RelatedEntityName	`String`	Yes	Specify the logical identifier of the related entity that can be retrieved.

LookUpContact<>

The LookUpContact<> activity retrieves the Id of contact records based on the value specified in the UserName parameter. This activity can be used in workflows that are specific to Forms Builder. Ensure that you do not use this activity in other workflows.

The retrieved ID serves as an input parameter in activities such as GetEntity<> or SaveEntity<>.



Cmc.NexusCrm.Common.Workflow.LookUpContact

ContactId	ContactID
DisplayName	LookUpContact
UserName	"Mark"

Properties

LookUpContact<> Properties

Property	Value	Required	Notes
ContactId	`InOutArgument<Int32>`	Yes	Specify the Id of the Contact that will be retrieved in the activity using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
UserName	`InArgument<String>`	Yes	Specify the registered user name of the student using a VB expression or variable.

Add a Lead

Matt Grammer is applying to the Engineering Department of Northside School of Engineering to pursue an undergraduate program in electrical engineering. When Matt submits his details on the website, a lead record is automatically created. Additionally, the associated contact record will be implicitly created by CampusNexus CRM.

1. Launch Workflow Composer.
2. Click **New Event Workflow**.
3. In the Name field, type a name for the workflow, e.g., **CreateLead**.
4. In the Entities area:
   a. Click ▶ next to **Cmc.NexusCrm.Common.Entities**
   b. Select **Void (VoidEntity)**. Select the entity which will trigger this workflow.
5. In the Events area, click **Saving (SavingEvent)**. Ensure that you select the appropriate event for the entity selected in the previous step.
6. Click **OK**. The sequence is created in the Designer pane.

Create an Entity

7. In the Toolbox, under Cmc.Core.Workflow.Activities.EntityModel, select the **CreateEntity<>** activity and drop it into the sequence. The Select Types window is displayed.
8. In the TEntity drop-down list, select **Browse for Types**. The Browse and Select a .Net Type window is displayed.
9. Select **Lead** and click **OK** twice. The Lead object is added to the CreateEntity<> activity in the sequence.
10. In the Variables pane, create a variable to hold the Lead instance object called **leadinstance**. In the Variable type field, select **Browse for type** and select **Cmc.NexusCrm.Common.Entities.Lead**.
11. In the Result field of the Properties pane, specify the name of the variable created above, e.g., **leadinstance**.

Assign Values to the Lead's Properties

12. From the Toolbox, drop an **Assign** activity into the sequence.
    a. In the To field, type the name of the variable created earlier (**leadinstance**) and append the lead's **Name** property to the variable.
    b. Type the name of the lead as **"Matt Grammer"**.
13. To assign values to the lead's email address, gender, campus, and team properties, perform the **Assign** operation as described in the previous step. Type the following values for each property:

To	Value
leadinstance.Email	“Mattg@mail.com”
leadinstance.Gender	1
leadinstance.Campus	1
leadinstance.Team	3

Associate a Related Entity to the Created Entity

14. Prior to associating the lead with an ethnic group record, create a variable for the ethnic group to be associated with the lead, e.g., **ethnicGroup**, and select **Com.NexusCrm.Common.Entities.Link** under Variable type.

15. In the Default column, type the value **New Link()**. The variable is created.

16. Perform the **Assign** operation again and add the following details:

To	Value
ethnicGroup.Id	5

This step adds the Id of the ethnic group that will be associated with the lead.

17. Perform the **Assign** operation to initialize the ethnic group collection in the lead instance. Type the following details:

To	Value
leadInstance.EthnicGroup	New EntityCollection(Of Link)

18. To add the ethnic group created in step 17 to the ethnic group collection, drop the **AddToCollection<>** activity into the Designer pane.

19. Type or select the following details in the Properties tab:

Property	Value
Collection	leadInstance.EthnicGroups
Item	ethnicGroup
TypeArgument	Cmc.NexusCrm.Common.entities.Link

20. From Cmc.Core.workflow.Activities.EntityModel in the Toolbox, drag the **SaveEntity<>** activity to the Designer pane. The Select Types dialog is displayed.

21. In the TEntity dropdown list, select **Cmc.Nexus.Crm.Common.Entities.Lead** and click **OK**.

22. In the **Properties** area, type the following values:

- DisplayName – type an appropriate display name.
- Entity – select **leadInstance**.
- ValidationMessages – this field is optional.

23. Click **Publish**. The New Workflow Definition Version window is displayed.

24. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.

25. Click **Save**, then **Cancel** to close the publisher window.

Add Attachments to a Contact Record

To complete his admission formalities, Matt Grammer, a lead at Northside School of Engineering, attaches copies of recommendation letters and previous education grades in his email to the university. When Matt sends these details, the attachments are automatically added to the Attachments tab of Matt’s contact record.

1. Launch Workflow Composer.
2. Click **New Event Workflow**.
3. In the Name field, type a name for the workflow (e.g., **AddingAttachment**)
4. In the Entities area:
    a. Click ▶ next to **Cmc.Core.Eventing**.
    b. Select **Void (VoidEntity)**. Select the appropriate entity for which the workflow must be triggered.
5. In the Events area, click **Saving (SavingEvent)**. In this step, ensure that you select the appropriate event for the entity selected in the previous step.
6. Click **OK**. The sequence is created in the Designer pane.

Retrieve the Contact Entity and its Associated Previous Education Records

7. In the Variables pane, create a variable for the contact object, e.g., **contact**, and select **Cmc.NexusCrm.Common.Entities.Contact** in the Variable type column
8. In the Toolbox, under Cmc.Core.Workflow.Activities.EntityModel, select the **GetEntity<>** activity and drop it into the Designer pane. The Select Types dialog box is displayed.
    a. In the TEntity drop-down list, select **Browse for Types**. The Browse and Select a .Net Type window is displayed.
    b. In the Type Name field, type **Contact**. The Contact object is selected under Cmc.NexusCrm.Common.Entities.
    c. Click **OK** twice. The Contact object is added to the GetEntity<> activity in the sequence.
    d. In the Toolbox, specify the entity identifier in the **EntityId** field.
    e. In the Result field, type the name of the variable created previously (**contact**). This entity will be retrieved in this workflow activity.
9. In the Toolbox, under Cmc.NexusCrm.Common.Workflow, select the **GetRelatedEntity<>** activity and drop it into the Designer pane. The Select Types dialog box is displayed.
    a. In the TEntity drop-down list, select **Cmc.NexusCrm.Common.Entities.Contact** and click **OK**.
    b. In the Toolbox, select the parent entity in the ParentEntity field, e.g., **contact**.

c. In the Type Name field, type **Contact**. The Contact object is selected under Cmc.NexusCrm.Common.Entities.

d. In the RelatedEntityName field, type the name of the related tab that needs to be fetched, e.g., **ContactPreviousEducations**.

Create a New Previous Education Record

10. In the Variables pane, create a variable for the lead, e.g., **previousEducation**, and select **Cmc.NexusCrm.Common.Entities.ContactPreviousEducation**.

11. In the Toolbox, under Cmc.Core.Workflow.Activities.EntityModel, select the **CreateEntity<>** activity and drop it into the sequence. The Select Types window is displayed.

    a. In the TEntity drop-down list, select **Browse for Types**. The Browse and Select a .Net Type window is displayed.

    b. Select **ContactPreviousEducation** and click **OK** twice. The Lead object is added to the CreateEntity<> activity in the sequence.

    c. In the Result field of the Properties pane, specify the name of the variable created above, e.g., **previousEducation**.

Assign Relationship Property Values to the Previous Education Record

12. From the Toolbox, drop an **Assign** activity for each row in the following table and type the indicated values:

To	Value
previousEducation.ContactPreviousEducationId	1
previousEducation.Gpa	4

In this step, the details of the new previous education record are set. The contact will be associated with the account instance assigned to previouEducation.ContactPreviousEducationId.

13. To add the previous education record to the previous education collection, drop the **AddToCollection<>** activity into the Designer pane.

14. Type or select the following details in the **Properties** tab:

    - Collection — AddToCollection<ContactPreviousEducation>
    - Item — previousEducation
    - TypeArgument — Cmc.NexusCrm.Common.entities.ContactPreviousEducation

Retrieve Attachments of the Contact Record

15. In the Toolbox, under Cmc.NexusCrm.Common.Workflow, select the **GetRelatedEntity<>** activity and drop it into the Designer pane. The Select Types dialog box is displayed.

a. In the TEntity drop-down list, select **Browse for Types**. The Browse and Select a .Net Type window is displayed.

b. In the Type Name field, type **Contact**. The Contact item is selected in Cmc.NexusCrm.Common.Entities.

c. Click **OK** twice.

d. In the Toolbox, select the parent entity in the ParentEntity field, e.g., **contact**.

e. In the Type Name field, type **Contact**. The Contact object is selected under Cmc.NexusCrm.Common.Entities.

f. In the RelatedEntityName field, type the name of the related tab that needs to be fetched, e.g., **“Attachments”**.

Set Attachment File Name and File Content

16. In the Variables pane, create a variable called **marksAttachment**. In the Variable type column, select **Cmc.NexusCrm.Common.Entities.Attachment**.

17. In the Default column, type **new Attachment()**.

18. From the Toolbox, drop an **Assign** activity for each row in the following table and type the indicated values:

To	Value
marksAttachment.FileName	“School Marks.doc”
marksAttachment.FileBlob	System.IO.File.ReadAllBytes(“<path of the School Marks.doc file>”)

Add the Attachment to the Retrieved Contact Record

19. To add the School Marks.doc file to the Attachment tab, drop the **AddToCollection<>** activity into the Designer pane.

20. Type or select the following details in the **Properties** tab:

    - Collection — contact.Attachments
    - DisplayName — AddToCollection<Attachment>
    - Item — marksAttachment
    - TypeArgument — Cmc.NexusCrm.Common.Entities.Attachment

21. From **Cmc.Core.Workflow.Activities.EntityModel** in the Toolbox, drag the **SaveEntity<>** activity to the Designer pane. The Select Types dialog is displayed.

22. In the TEntity drop-down list, select **Cmc.Nexus.Crm.Common.Entities.Contact** and click **OK**.

23. In the **Properties** area, type the following values:

- DisplayName — type an appropriate display name.
- Entity — type contact.
- ValidationMessages — this field is optional.

24. Click **Publish**. The New Workflow Definition Version window is displayed.
25. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.
26. Click **Save**, then **Cancel** to close the publisher window.

Register Participants

This sample workflow demonstrates how to register participant(s) using a workflow. This sample demonstrates how to register a lead record as a participant.

Prerequisite

Event support should be enabled for the Entity from Business Administrator. For information about enabling Event support, see CampusNexus CRM Business Administrator Help.

Business Flow

1. A participant record is created when an instance of the object for which Event support is enabled is added to an event.
2. When a participant is registered for an event through workflow:
   - For a paid event (Event.EventType=”Paid”), the participant status is marked as **Pending** until the participant pays for the event.

     The participant is blocked to attend the event for a limited duration (configurable through the Talisma-ClearBlockedParticipant 7EE38D20-D097-11d2-BE17-00C04FCCE602 <database name> job. If the money is not paid within the duration, the participant’s status will be set to Payment failed, and the participant will be cleared from the blocked state.
   - The user must explicitly call the **UpdateEntity<>** of the Participant Object to update the status for the added participants.
   - For a free event, the participant status is set implicitly to **Registered**.
3. The Available Seats Calculation is based on Event.ParticipantLimit – Number of participants blocked for the event.

Notes:

- The **Allow registration for the series** property is not applicable in workflows. During event registration, participants will be added only to the main event and not to any sub-events.
- For object instances other than instances of the Contact object, during event registration the Primary Participant entity (Lead or custom object) must be associated with a contact. Event registration will fail if the association is not created.

Register Lead Entities in an Event

1. In the Variables pane, create a variable for the Lead Service, e.g., **Leadsvc**.

   In the Variable type column, select **Cmc.NexusCrm.Common.Services.ILeadService**.
2. In the Toolbox, under Cmc.Core.Workflow.Activities, select the **GetServiceInstance<>** activity and drop it into the Designer pane. The Select Types dialog box is displayed.

   a. In the TService drop-down list, select **Browse for Types**. The Browse and Select a .Net Type window is displayed.

   b. In the Type Name field, type **ILeadService**. The ILeadService is selected under Cmc.NexusCrm.Common.Services.

   c. Click **OK** twice. The ILeadService is added to the GetServiceInstance<> activity in the sequence.

   d. In the Result field, type the name of the variable created previously (**Leadsvc**). This service will be retrieved in this workflow activity.

3. In the Variables pane, create a variable (e.g., **request**).

   In the Variable type column:

   a. Select **Browse for Types**. The Browse and Select a .Net Type window is displayed.

   b. In the Type Name field, type **RegisterParticipantRequest<T>** under Cmc.NexusCrm.Core.Contracts.Services.Common and select **Cmc.NexusCrm.Common.Entities.Lead** from the T drop-down.

   c. In the Default column, specify **new RegisterParticipantRequest(of Lead)**.

4. In the Toolbox, under Primitives, select the **Assign** activity and drop it into the current workflow sequence.

   a. In the To field, specify **request.EventID**.

   b. In the Value field, specify **<Id of the Event>**.

Add a Primary Participant to the Event

The primary participant (lead) can be passed from a form or retrieved through a workflow. If it’s passed from a form, pass the argument name **lead** as the primary participant parameter to the above method. If it has to be retrieved from the system, based on business requirements, use the GetEntity<> activity to retrieve the lead.

1. In the Toolbox, under Primitives, select the **Assign** activity and drop it into the workflow sequence.

   a. In the To field, specify **request.PrimaryParticipant**.

   b. In the Value field, specify **lead** (Either the variable/argument as appropriate).

Add a Secondary Participant to the Event

In this example, to set the secondary participant to the request, data will be retrieved from the system using the GetEntity<> activity. However, data can also be retrieved from a form.

1. In the Variables pane, create another variable for the Lead object, e.g., **LeadSecondary**.

   In the Variable type column, select **Cmc.NexusCrm.Common.Entities.Lead**.

2. In the Toolbox, under Cmc.Core.Workflow.Activities.EntityModel, select the **GetEntity<>** activity and drop it into the Designer pane. The Select Types dialog box is displayed:

    a. In the TService drop-down list, select **Browse for Types**. The Browse and Select a .Net Type window is displayed.

    b. In the Type Name field, type **Lead**. The Lead object is selected under Cmc.NexusCrm.Common.Entities.

    c. Click **OK** twice. The Contact object is added to the GetEntity<> activity in the sequence.

    d. In the EntityId field, specify another Lead Id (the lead you want to register in the event).

    e. In the Result field, type the name of the second variable (**LeadSecondary**). This entity will be retrieved in this workflow activity.

3. In the Variables pane, create a variable for a collection of the lead entity, e.g., **LeadCollection**.

    In the Variable type column:

    a. Select **Browse for Types**. The Browse and Select a .Net Type window is displayed.

    b. In the Type Name field, type **Collection<T>** under System.Collections.ObjectModel and select **Cmc.NexusCrm.Common.Entities.Lead** from the T drop-down.

    c. In the Default column, specify **new RegisterParticipantRequest(of Lead)**.

4. In the Toolbox, under Collection, select the **AddToCollection<>** activity and drop it into the workflow sequence.

    In the right pane:

    a. In the Collection field, specify **LeadCollection**.

    b. In the Item field, specify **LeadSecondary**.

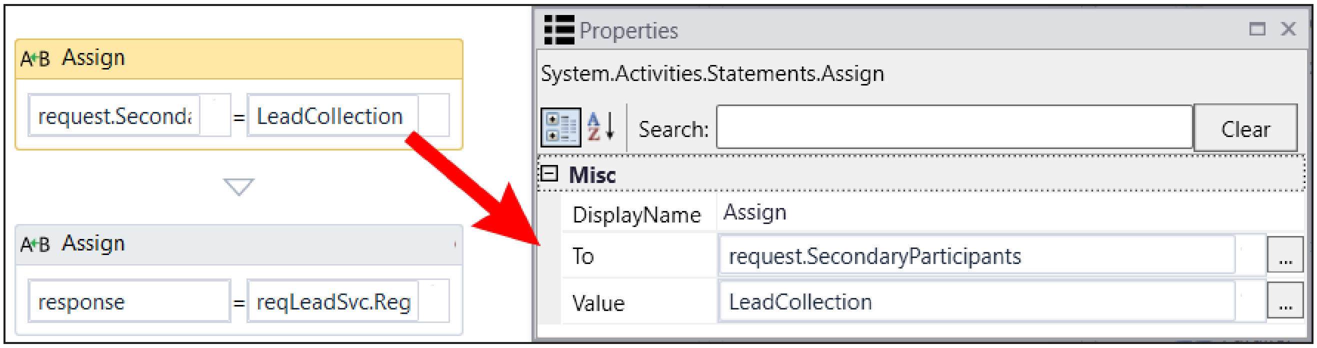
    c. In the TypeArgument field, specify **Cmc.NexusCrm.Common.Entities.Lead**.

    Note: Repeat steps 2 and 4 of the previous procedure for each Lead instance you want to register in the event as part of group registration.

5. In the Toolbox, under Primitives, select the **Assign** activity and drop it into the workflow sequence.

    a. In the To field, specify **request.SecondaryParticipants**.

    b. In the Value field, specify **LeadCollection**.



A+B Assign
request.Second = LeadCollection

A+B Assign
response = reqLeadSvc.Reg

System.Activities.Statements.Assign

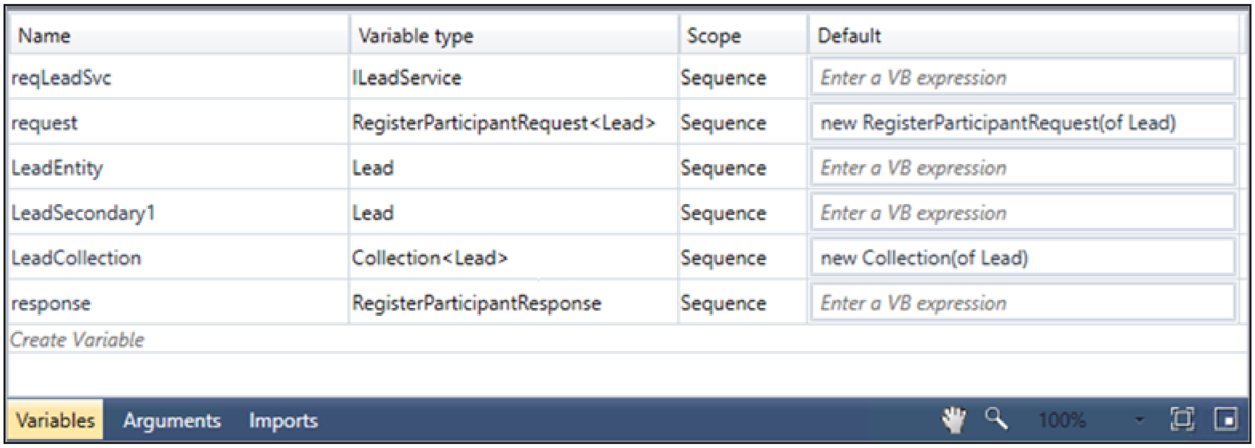
DisplayName	Assign
To	request.SecondaryParticipants
Value	LeadCollection

6. In the Variables pane, create a variable e.g., **response**.

   In the Variable Type column:

   a. Select **Browse for Types**. The Browse and Select a .Net Type window is displayed.

   b. In the Type Name field, type **RegisterParticipantResponse** under Cmc.NexusCrm.Core.Contracts.Services.Common.

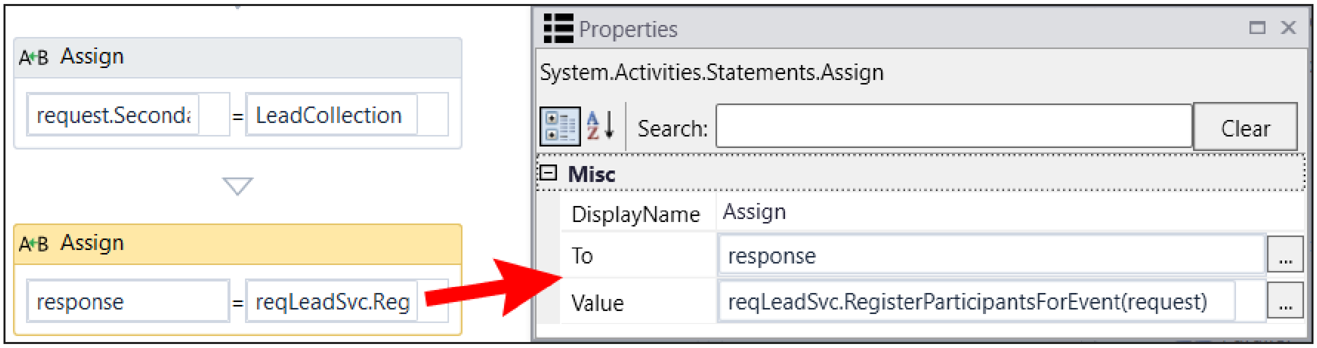


Name	Variable type	Scope	Default
reqLeadSvc	ILeadService	Sequence	*Enter a VB expression*
request	RegisterParticipantRequest<Lead>	Sequence	new RegisterParticipantRequest(of Lead)
LeadEntity	Lead	Sequence	*Enter a VB expression*
LeadSecondary1	Lead	Sequence	*Enter a VB expression*
LeadCollection	Collection<Lead>	Sequence	new Collection(of Lead)
response	RegisterParticipantResponse	Sequence	*Enter a VB expression*
Create Variable			

7. In the Toolbox, under Primitives, select the **Assign** activity and drop it into the workflow sequence.

   a. In the To field, specify **response**.

   b. In the Value field, specify **reqLeadSvc.RegisterParticipantsForEvent(request)**.



A+B Assign
request.Second = LeadCollection

A+B Assign
response = reqLeadSvc.Reg

System.Activities.Statements.Assign

DisplayName	Assign
To	response
Value	reqLeadSvc.RegisterParticipantsForEvent(request)

Check for Duplicate Records

You can create a workflow to filter the creation of duplicate records for entities that are available in OData. This functionality can be achieved through the ExecuteODataQuery<> activity.

Business Scenario

An institution wants to prevent the creation of new leads as lead records are already available in the database. The filter criteria to check for a duplicate lead can be:

- Firstname and lastname and email and mobile
  — OR —
- Firstname and lastname and email
  — OR —
- Firstname and lastname and mobile
  — OR —
- Email and mobile

Create a Workflow With the Above Logic

1. Declare the following variables in the order in which they are specified and include their indicated values:

Variable	Type	Scope	Value
baseODataUrl	String	Sequence	https://<Forms Builder Renderer URL>/api/ApiProxy/CRM/
dupCheckEntity	String	Sequence	"Leads?" **Note**: For a different object, replace this string with the name of the object as it appears in OData.
oDataSelectClause	String	Sequence	"$select=LeadId" **Note**: For a different object, replace this value with the identifier of the object.
firstCondition (First condition in the business scenario)	String	Sequence	"FirstName eq '" & lead.FirstName & "' and LastName eq '" & lead.LastName & "' and Email eq '" & lead.Email & "' and Mobile eq '"& lead.Mobile & "' "
secondCondition (Second condition in the business scenario)	String	Sequence	"FirstName eq '" & lead.FirstName & "' and LastName eq '" & lead.LastName & "' and Email eq '" & lead.Email & "' "
thirdCondition (Third condition in the business scenario)	String	Sequence	"FirstName eq '" & lead.FirstName & "' and LastName eq '" & lead.LastName & "' and Mobile eq '" & lead.Mobile & "' "
fourthCondition (Fourth condition in the business scenario)	String	Sequence	"Email eq '" & lead.Email & "' and Mobile eq '" & lead.Mobile & "' "
oDataFilterClause (Collates all the filter conditions)	String	Sequence	"$filter=(" & firstCondition & " or " & secondCondition & " or " & thirdCondition & " or " & fourthCondition & ")"
oDataOrderbyClause (Will list the most recently created duplicate lead record)	String	Sequence	"$orderby=CreatedOn desc"
segmentTerminator	String	Sequence	"&"
oDataQuery (Finally constructed OData query)	String	Sequence	baseODataUrl & dupCheckEntity & oDataSelectClause & segmentTerminator & oDataFilterClause & segmentTerminator & oDataOrderbyClause

Note: Highlighted elements must be replaced appropriately if the base object is not Lead.

2. Add the **CreateEntity<Lead>** activity in the Entry section.

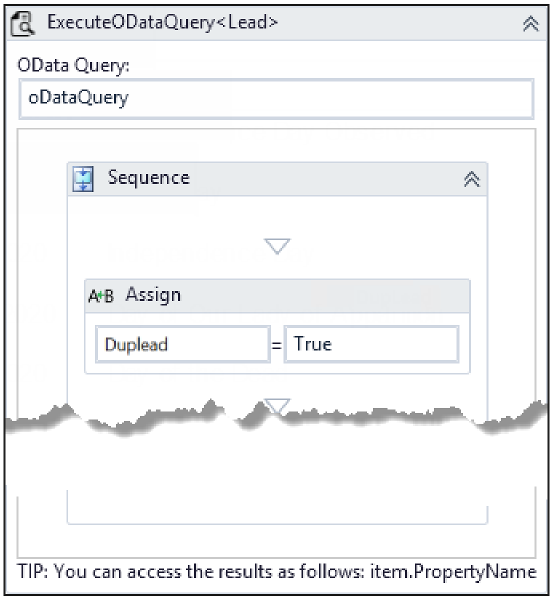
3. Declare a **Boolean variable** (for example, Duplead) and set its default value to **False**.

4. In the Transition(s) section, click **Next** and add the **ExecuteODataQuery<Lead>** activity.

5. In the OData Query field, type **oDataQuery**.

6. Add a **Sequence** to the ExecuteODataQuery<Lead> activity which includes logic to identify if a duplicate lead is found.

   For example:



ExecuteODataQuery<Lead>

OData Query: oDataQuery

Sequence

Assign

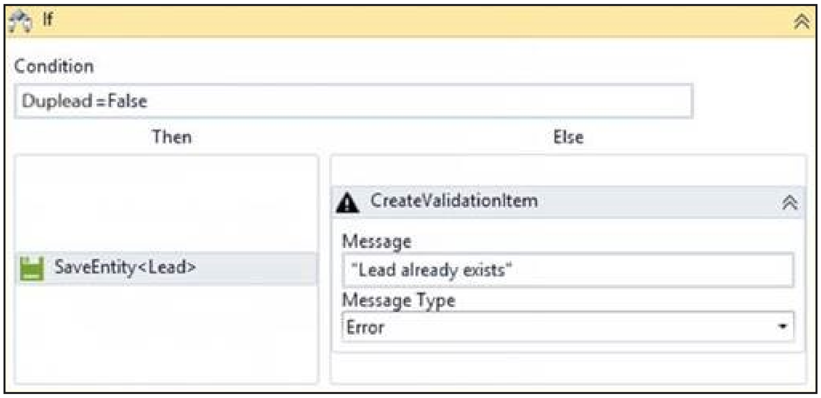
Duplead = True

TIP: You can access the results as follows: item.PropertyName

7. Include an **If condition** with the following logic:

   - If the value of the Duplead flag is unchanged, a new Lead record will be created.
   - If a duplicate lead is found, the value of the Duplead flag will be changed to True and the included validation message "*Lead already exists*" will be displayed.

   For example:



If

Condition: Duplead=False

Then: SaveEntity<Lead>

Else: CreateValidationItem

Message: "Lead already exists"

Message Type: Error

Activities for Anthology Student

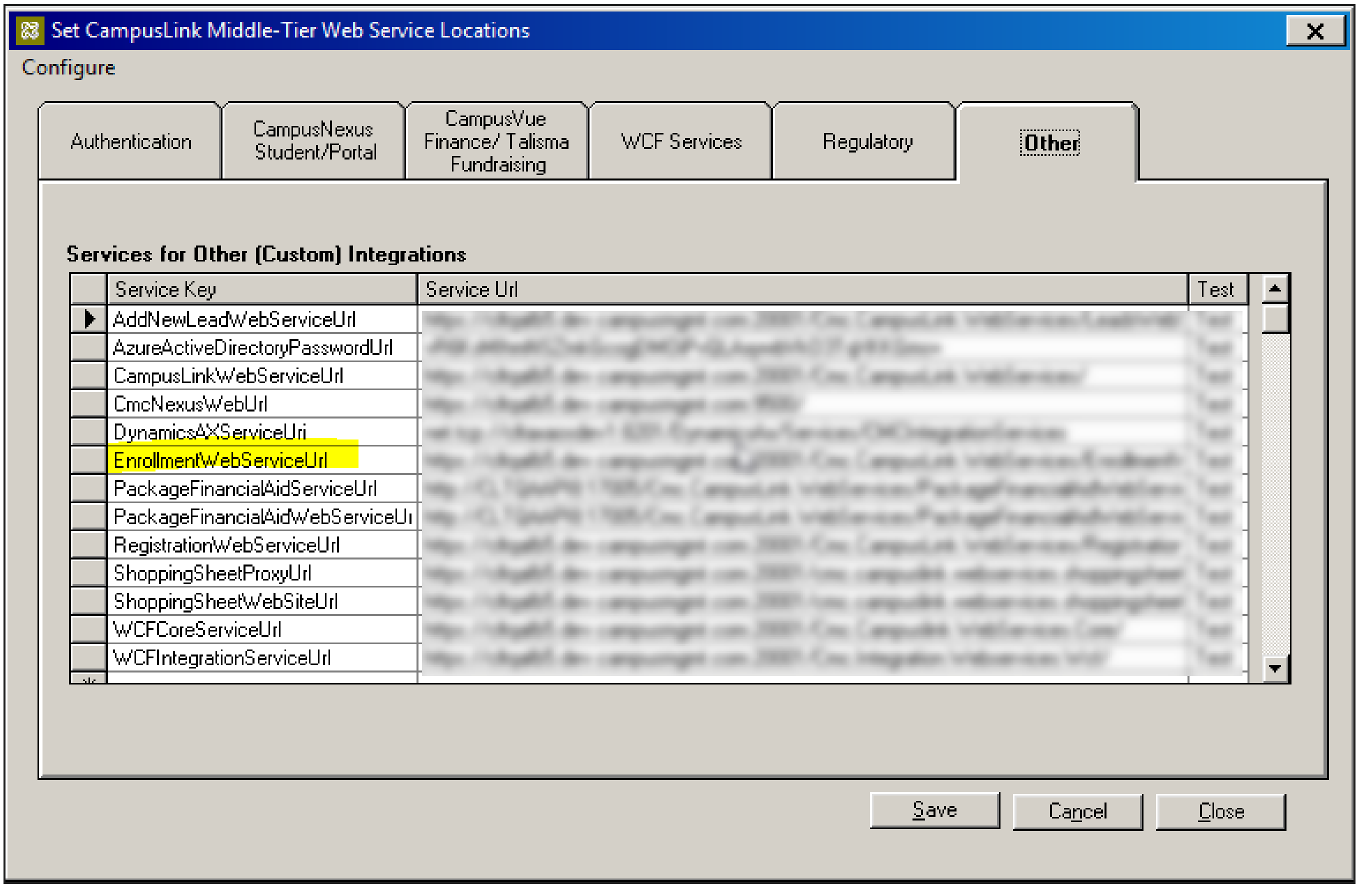
The activities in this section are designed for use with Anthology Student.

ConvertApplicantToEnrollment (V2)

The ConvertApplicantToEnrollment activity enables you to promote an Applicant record to an Enrollment and invoke the enrollment business logic.

If you use the ConvertApplicantToEnrollment (V2) activity with Anthology Student Activities and Contracts (**V1**) (instead of V2), the **EnrollmentWebServiceUrl** key needs to be set.

To verify this in the Desktop client, navigate to **Setup > System > File Server & Services**, click **Configure Web Service Locations**, and select the **Other** tab.

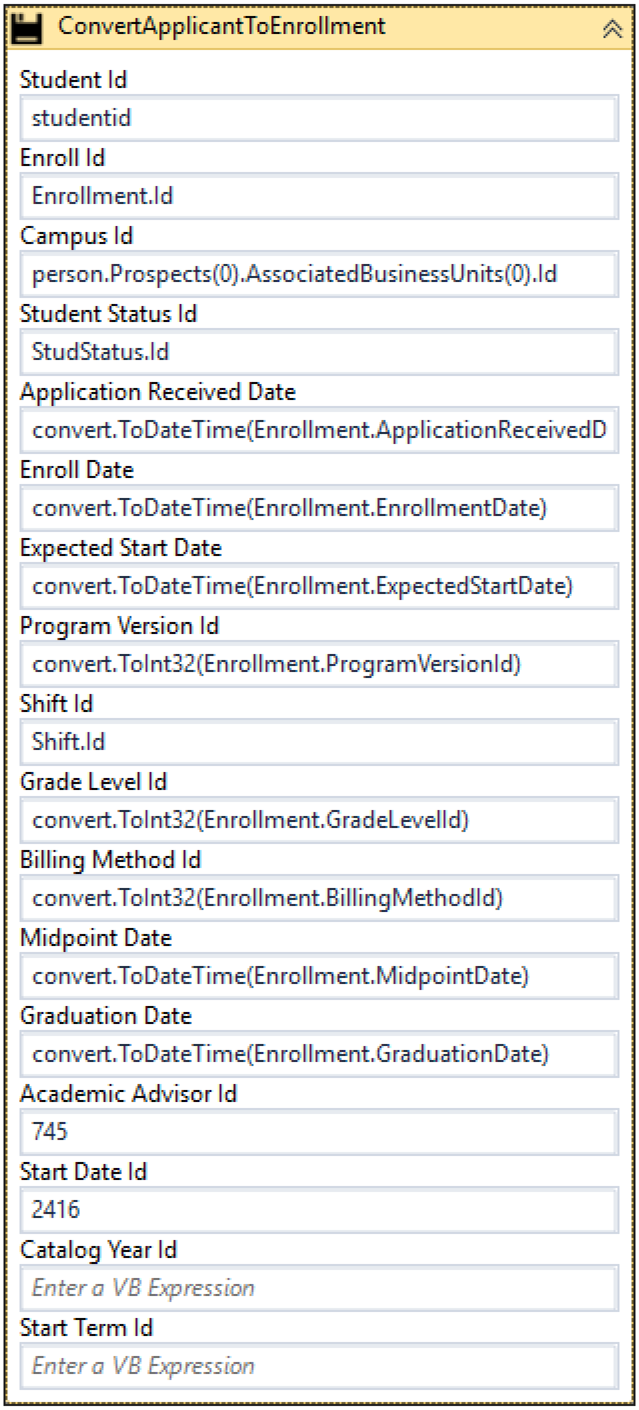

Set CampusLink Middle-Tier Web Service Locations
Configure
Authentication
CampusNexus Student/Portal
CampusVue Finance/ Talisma Fundraising
WCF Services
Regulatory
Other
Save
Cancel
Close

Services for Other (Custom) Integrations

Service Key	Service Url	Test
AddNewLeadWebServiceUrl	[redacted]	Test
AzureActiveDirectoryPasswordUrl	[redacted]	Test
CampusLinkWebServiceUrl	[redacted]	Test
CmcNexusWebUrl	[redacted]	Test
DynamicsAXServiceUri	[redacted]	Test
EnrollmentWebServiceUrl	[redacted]	Test
PackageFinancialAidServiceUrl	[redacted]	Test
PackageFinancialAidWebServiceUr	[redacted]	Test
RegistrationWebServiceUrl	[redacted]	Test
ShoppingSheetProxyUrl	[redacted]	Test
ShoppingSheetWebSiteUrl	[redacted]	Test
WCFCoreServiceUrl	[redacted]	Test
WCFIntegrationServiceUrl	[redacted]	Test

Use Case

When an online application is submitted through Forms Builder, an Applicant record is created that leverages the student statuses associated with the configured Applicant Category statuses. Once the student is approved for enrollment, typically the Anthology Student user would use the Enrollment Wizard and populate the enrollment with the existing information. Instead of having a user go through the Enrollment Wizard, a workflow can detect an approval from a Contact Manager activity, Document Status event, or Group Membership event and then, using the ConvertApplicantToEnrollment activity promote the applicant record to a full enrollment.



ConvertApplicantToEnrollment

Student Id

studentid

Enroll Id

Enrollment.Id

Campus Id

person.Prospects(0).AssociatedBusinessUnits(0).Id

Student Status Id

StudStatus.Id

Application Received Date

convert.ToDateTime(Enrollment.ApplicationReceivedD

Enroll Date

convert.ToDateTime(Enrollment.EnrollmentDate)

Expected Start Date

convert.ToDateTime(Enrollment.ExpectedStartDate)

Program Version Id

convert.ToInt32(Enrollment.ProgramVersionId)

Shift Id

Shift.Id

Grade Level Id

convert.ToInt32(Enrollment.GradeLevelId)

Billing Method Id

convert.ToInt32(Enrollment.BillingMethodId)

Midpoint Date

convert.ToDateTime(Enrollment.MidpointDate)

Graduation Date

convert.ToDateTime(Enrollment.GraduationDate)

Academic Advisor Id

745

Start Date Id

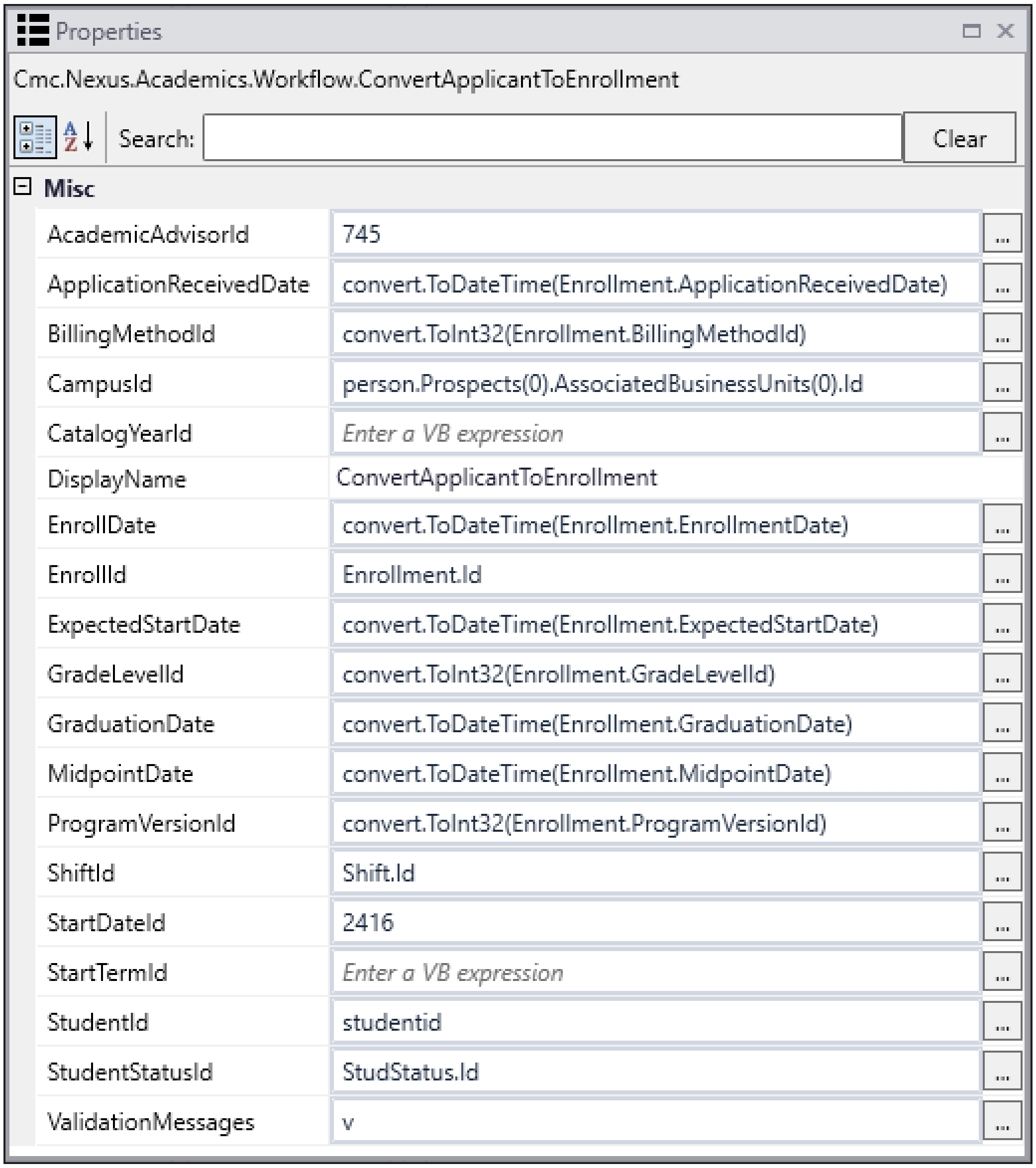
2416

Catalog Year Id

Enter a VB Expression

Start Term Id

Enter a VB Expression



Cmc.Nexus.Academics.Workflow.ConvertApplicantToEnrollment

Misc

AcademicAdvisorId	745
ApplicationReceivedDate	convert.ToDateTime(Enrollment.ApplicationReceivedDate)
BillingMethodId	convert.ToInt32(Enrollment.BillingMethodId)
CampusId	person.Prospects(0).AssociatedBusinessUnits(0).Id
CatalogYearId	*Enter a VB expression*
DisplayName	ConvertApplicantToEnrollment
EnrollDate	convert.ToDateTime(Enrollment.EnrollmentDate)
EnrollId	Enrollment.Id
ExpectedStartDate	convert.ToDateTime(Enrollment.ExpectedStartDate)
GradeLevelId	convert.ToInt32(Enrollment.GradeLevelId)
GraduationDate	convert.ToDateTime(Enrollment.GraduationDate)
MidpointDate	convert.ToDateTime(Enrollment.MidpointDate)
ProgramVersionId	convert.ToInt32(Enrollment.ProgramVersionId)
ShiftId	Shift.Id
StartDateId	2416
StartTermId	*Enter a VB expression*
StudentId	studentid
StudentStatusId	StudStatus.Id
ValidationMessages	v

Properties

ConvertApplicantToEnrollment Properties

Property	Value	Required	Notes
AcademicAdvisorId	`InArgument<Nullable><Int32>>`	No*	Specify the Academic Advisor Id using a VB expression or variable. *** Note**: The Academic Advisor Id is required or optional depending on a setting in Anthology Student: • The Academic Advisor Id is optional when 'Advisor Selection' is cleared under Setup > Academic Records > Enrollment. • The Academic Advisor Id is required when 'Advisor Selection' is selected under Setup > Academic Records > Enrollment.
ApplicationReceivedDate	`InArgument<DateTime>`	Yes	Specify the date when the student's application was received using a VB expression or variable.
BillingMethodId	`InArgument<Int32>`	Yes	Specify the database identifier for the Billing Method using a VB expression or variable.
CampusId	`InArgument<Int32>`	Yes	Specify the database identifier for the Campus in which the student is enrolled using a VB expression or variable.
CatalogYearId	`InArgument<Nullable><Int32>`	No	Specify the catalog year identifier using a VB expression or variable. **Note**: This property is available only in the V2 version of the activity, i.e., in the Cmc.Nexus.Academics.Workflow namespace.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EnrollDate	`InArgument<DateTime>`	Yes	Specify date when the student is enrolled into the Program using a VB expression or variable.

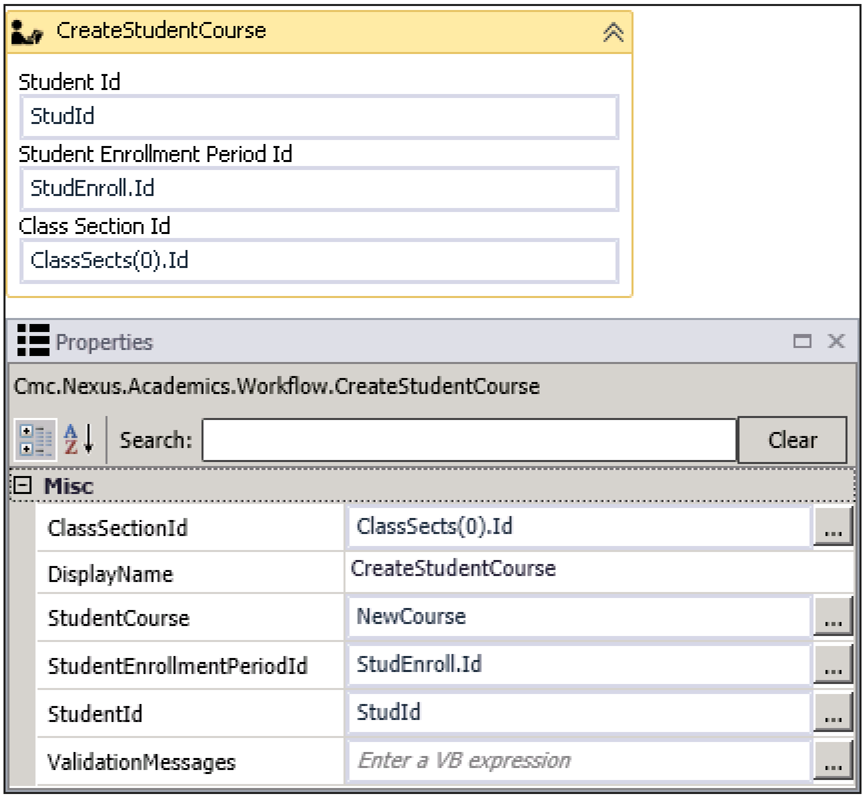
Property	Value	Required	Notes
EnrollId	`InArgument<Int32>`	Yes	Specify the Student Enrollment Period Id using a VB expression or variable.
ExpectedStartDate	`InArgument<DateTime>`	Yes	Specify the date that the student is expected to start using a VB expression or variable.
GradeLevelId	`InArgument<Int32>`	Yes	Specify the database identifier for Grade Level for this enrollment using a VB expression or variable.
GraduationDate	`InArgument<DateTime>`	No	Specify the Graduation Date using a VB expression or variable.
MidpointDate	`InArgument<DateTime>`	No	Specify the Midpoint Date using a VB expression or variable.
ProgramVersionId	`InArgument<Int32>`	Yes	Specify the database identifier for Program Version for this enrollment using a VB expression or variable.
ShiftId	`InArgument<Int32>`	Yes	Specify the database identifier for the Shift from the AdShift table (Day, Night, etc.) using a VB expression or variable.

Property	Value	Required	Notes
StartDateId	`InArgument<Nullable><Int32>>`	No*	Specify the database identifier for the Start Date using a VB expression or variable. * **Note**: The Start Date Id is required or optional depending on settings in Anthology Student: Required: ◦ If "Require Start Date' is selected under Setup > Academic Records > Settings. ◦ If MidpointDate/GraduationDate are not specified. Optional: ◦ If 'Require Start Date' is set to 'Not Required' under Setup > Academic Records > Settings. ◦ If MidpointDate/GraduationDate are specified.
StartTermId	`InArgument<Nullable><Int32>`	No	Specify the start term year identifier using a VB expression or variable. **Note**: This property is available only in the V2 version of the activity, i.e., in the Cmc.Nexus.Academics.Workflow namespace.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

CreateStudentCourse (V2)

The CreateStudentCourse activity enables you to create a Student Course so that the student can be registered in that course.

This activity creates an instance of a Student Course; it does not save it to the database. The workflow can include other activities that manipulate the Student Course before it is saved. To persist the Student Course in the database, insert a SaveStudentCourse (V2) activity.



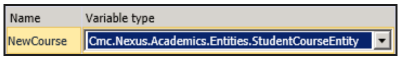
Student Id: StudId
Student Enrollment Period Id: StudEnroll.Id
Class Section Id: ClassSects(0).Id

ClassSectionId	ClassSects(0).Id
DisplayName	CreateStudentCourse
StudentCourse	NewCourse
StudentEnrollmentPeriodId	StudEnroll.Id
StudentId	StudId
ValidationMessages	Enter a VB expression

Properties

CreateStudentCourse Properties

Property	Value	Required	Notes
ClassSectionId	`InArgument<Int32>`	Yes	Specify the Class Section Id using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

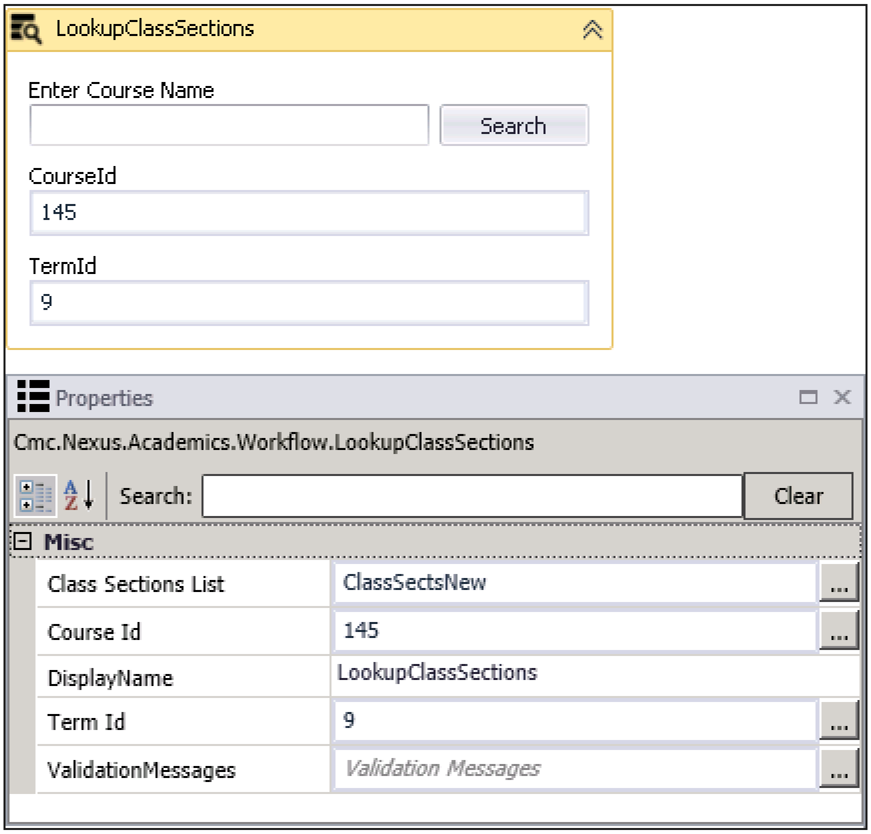
Property	Value	Required	Notes
StudentCourse	`OutArgument <StudentCourseEntity>`	Yes	The Student Course created by this workflow activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities**, select **StudentCourseEntity**, and click **OK**.  Name Variable type NewCourse Cmc.Nexus.Academics.Entities.StudentCourseEntity See StudentCourseEntity Class in the Anthology Student Object Library.
StudentEnrollmentPeriodId	`InArgument<Int32>`	Yes	Specify the Student Enrollment Period Id using a VB expression or variable.
StudentId	`InArgument<Int32>`	Yes	Specify the Student Id using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

LookupClassSections (V2)

The LookupClassSections activity is a lookup function that finds the Course Id for a class section based on a specified Course Name and Term Id. The activity includes a Search tool that returns Course Names and Course Codes. When you select a Course in the Search tool, the selected item is inserted into the Course Name field of the LookupClassSections activity and the Search tool is closed. You can use this lookup function during a course registration activity.

Use Case

A workflow detects when a student's status changes from any status to an enrolled status and automatically registers the student into an introductory course (Intro101). The LookupClassSections activity is used in the workflow to determine the Course Id (that is, the ClassSectionId of the StudentCourse) for the Intro101 course in the applicable term.



LookupClassSections

Enter Course Name

CourseId: 145

TermId: 9

Cmc.Nexus.Academics.Workflow.LookupClassSections

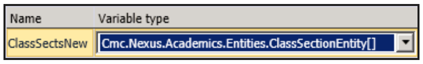
Search:

Misc

Class Sections List	ClassSectsNew
Course Id	145
DisplayName	LookupClassSections
Term Id	9
ValidationMessages	*Validation Messages*

Properties

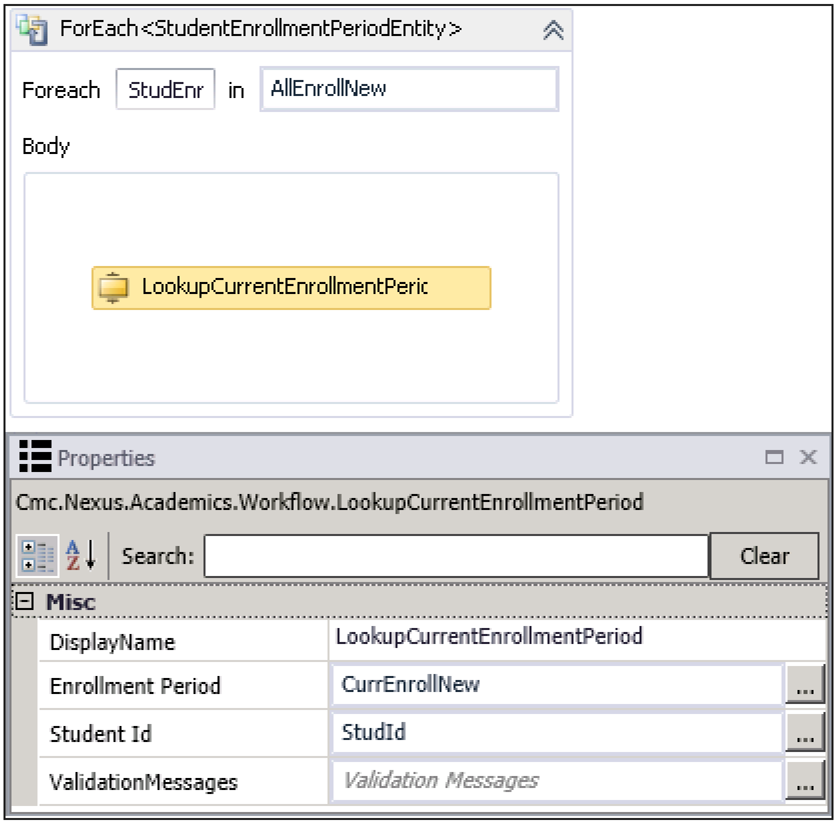
LookupClassSections Properties

Property	Value	Required	Notes
Class Section List	`OutArgument <ClassSectionEntity[]>`	Yes	The LookupClassSections activity returns an array of class sections associated with a course. Specify a course name in the Course Name field or click the Search button to find a course and select it. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities**, select **ClassSectionEntity**, and click **OK**.  Name Variable type ClassSectsNew Cmc.Nexus.Academics.Entities.ClassSectionEntity[] See ClassSectionEntity Class in the Anthology Student Object Library.
Course Id	`InArgument<Int32>`	Yes	The Course Id is a variable captured from an event.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Term Id	`InArgument<Int32>`	No	The Term Id is a variable captured from an event. The Terms property is a collection. See ClassSectionEntity.Terms Property in the Anthology Student Object Library.

Property	Value	Required	Notes
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

LookupCurrentEnrollmentPeriod (V2)

The LookupCurrentEnrollmentPeriod activity is a function that captures the Student Id from an event and returns the current enrollment period for the student. Use this lookup function when you need to know the current enrollment period in a workflow that has preceding activities containing the Student Id.


ForEach<StudentEnrollmentPeriodEntity>
Foreach
StudEnr
in
AllEnrollNew
Body
LookupCurrentEnrollmentPeric
Properties
Search:
Clear
Misc

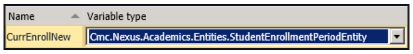
Cmc.Nexus.Academics.Workflow.LookupCurrentEnrollmentPeriod

DisplayName	LookupCurrentEnrollmentPeriod
Enrollment Period	CurrEnrollNew
Student Id	StudId
ValidationMessages	*Validation Messages*

Properties

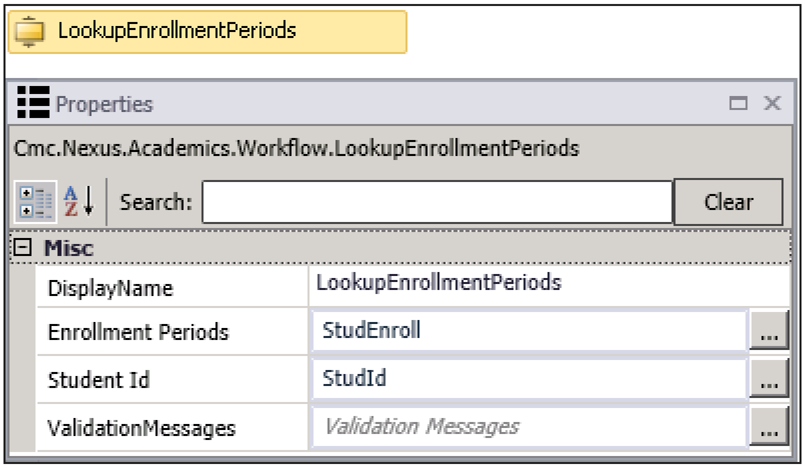
LookupCurrentEnrollmentPeriod Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
Student Enrollment Period	`OutArgument <StudentEnrollmentPeriodEntity>`	Yes	The current enrollment period returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities**, select **StudentEnrollmentPeriodEntity** and click **OK**.  Name / Variable type CurrEnrollNew / Cmc.Nexus.Academics.Entities.StudentEnrollmentPeriodEntity See StudentEnrollmentPeriodEntity Class in the Anthology Student Object Library.
Student Id	`InArgument<Int32>`	Yes	The Student Type Id captured from an event.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

LookupEnrollmentPeriods (V2)

The LookupEnrollmentPeriods activity is a function that captures the Student Id from an event and returns a list of all enrollment periods. Use this lookup function when you need to know the enrollment periods in a workflow that has preceding activities containing the Student Id.



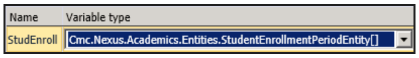
Cmc.Nexus.Academics.Workflow.LookupEnrollmentPeriods

DisplayName	LookupEnrollmentPeriods
Enrollment Periods	StudEnroll
Student Id	StudId
ValidationMessages	Validation Messages

Properties

LookupEnrollmentPeriods Properties

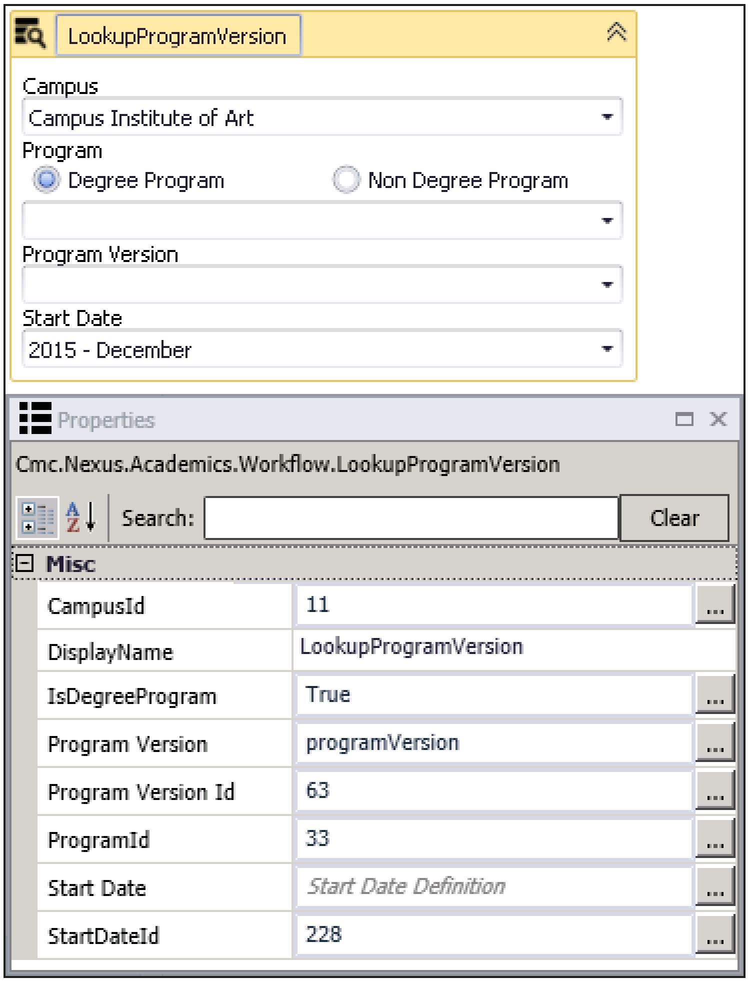
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
Student Enrollment Period	`OutArgument <StudentEnrollmentPeriodEntity[]>`	Yes	A list of all enrollment periods returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities**, select **StudentEnrollmentPeriodEntity**, and click **OK**.  Name | Variable type StudEnroll | Cmc.Nexus.Academics.Entities.StudentEnrollmentPeriodEntity[] See StudentEnrollmentPeriodEntity Class in the Anthology Student Object Library.
Student Id	`InArgument<Int32>`	Yes	The Student Id captured from an event.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

LookupProgramVersion

The LookupProgramVersion activity is a function that captures the Program Version Id, Campus Id, Program Id, and Start Date Id from an event and returns the Program Version. The lookup can be applied to Degree Programs or Non Degree Programs.

You can use this lookup function to retrieve a specific program version record when a new enrollment is saved from the workflow.



Campus: Campus Institute of Art

Program: ● Degree Program ○ Non Degree Program

Program Version:

Start Date: 2015 - December

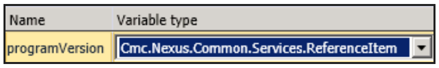
Cmc.Nexus.Academics.Workflow.LookupProgramVersion

CampusId	11
DisplayName	LookupProgramVersion
IsDegreeProgram	True
Program Version	programVersion
Program Version Id	63
ProgramId	33
Start Date	*Start Date Definition*
StartDateId	228

Properties

LookupProgramVersion Properties

Property	Value	Required	Notes
CampusId	`InArgument<Int32>`	No	Select a value in the drop-down list of the activity in the Designer window.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
IsDegreeProgram	`InArgument<Boolean>`	Yes	A Boolean expression that specifies whether the Program Version is associated with a Degree Program. The default value is false, that is, Non Degree Program.
Program Version	`OutArgument<ReferenceItem>`	Yes	The Program Version returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Common.Contracts > Cmc.Nexus.Common.Services**, select **ReferenceItem**, and click **OK**.  Name Variable type programVersion Cmc.Nexus.Common.Services.ReferenceItem
Program Version Id	`InArgument<Int32>`	Yes	This Id is populated by the activity based on your selections in the Campus Id, Program, Program Version, and Start Date fields.
ProgramId	`InArgument<Int32>`	No	This Id is populated by the activity based on your selections in the Campus Id, Program, Program Version, and Start Date fields.

Property	Value	Required	Notes
Start Date	`OutArgument<ReferenceItem>`	No	The Start Date returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Common.Contracts > Cmc.Nexus.Common.Services**, and select **ReferenceItem**.
StartDateId	`InArgument<Int32>`	No	This Id is populated by the activity based on your selections in the Campus Id, Program, Program Version, and Start Date fields.

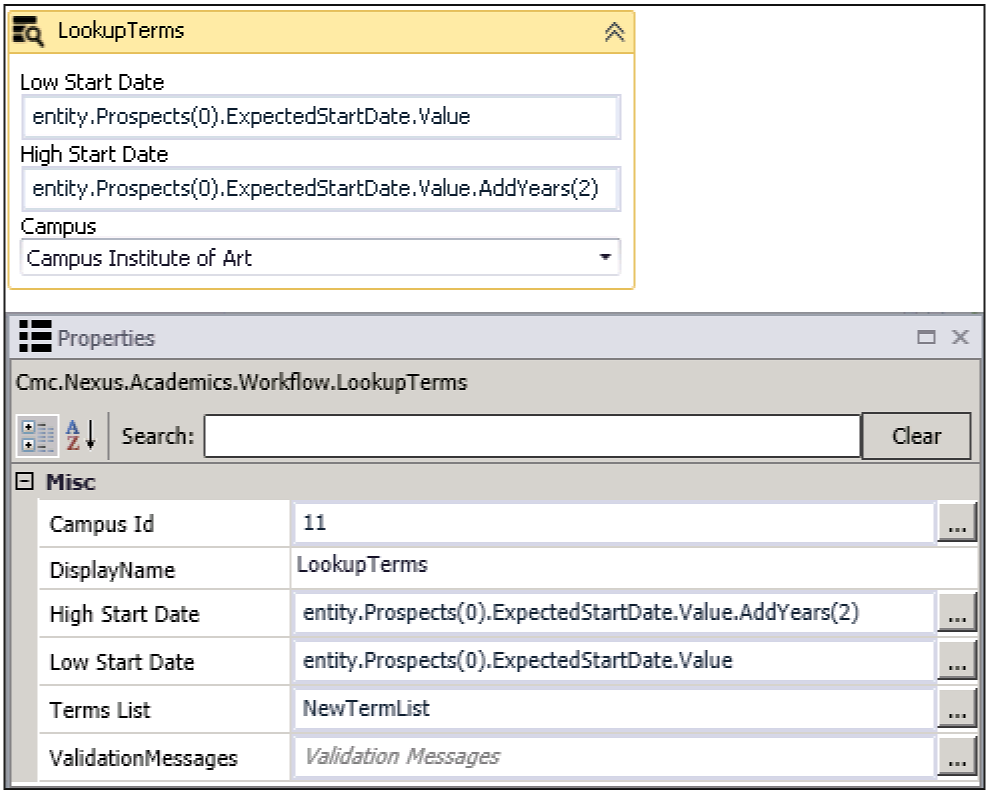
LookupTerms (V2)

The LookupTerms activity is a function that captures the Campus Id from an event and returns the Terms for a specified time period.

Use Cases

You could use this activity in a workflow on a Saving event since the Expected Start Date is entered on the Student Master form. The workflow could check whether a valid term start date is entered and provide a validation message.

Another way to use LookupTerms is to create a workflow with a ForEach loop that lists Term start dates within a certain time period of Expected Start Date. The list of Term start dates could be displayed in an Information message.



LookupTerms

Low Start Date: entity.Prospects(0).ExpectedStartDate.Value

High Start Date: entity.Prospects(0).ExpectedStartDate.Value.AddYears(2)

Campus: Campus Institute of Art

Properties

Cmc.Nexus.Academics.Workflow.LookupTerms

Search:

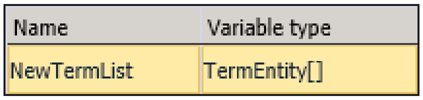
Misc

Campus Id	11
DisplayName	LookupTerms
High Start Date	entity.Prospects(0).ExpectedStartDate.Value.AddYears(2)
Low Start Date	entity.Prospects(0).ExpectedStartDate.Value
Terms List	NewTermList
ValidationMessages	*Validation Messages*

Properties

LookupTerms Properties

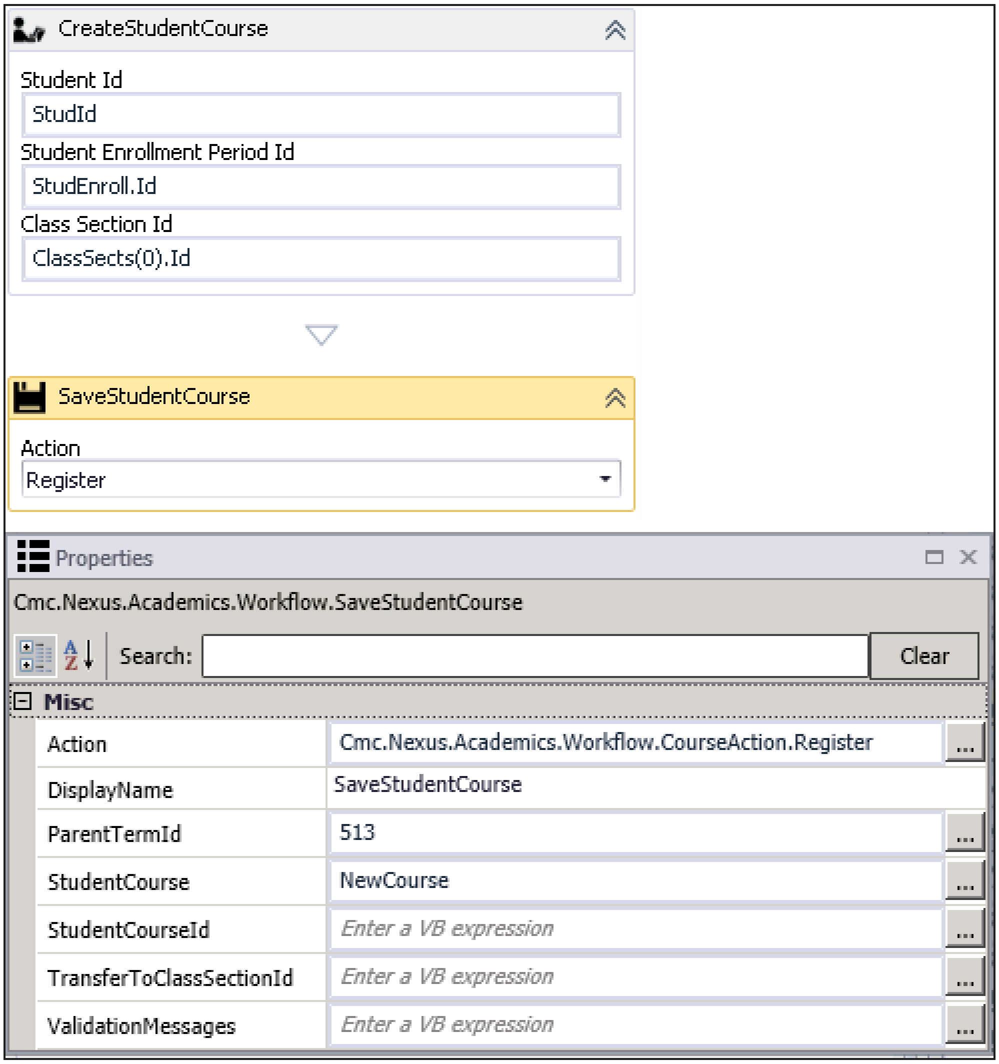
Property	Value	Required	Notes
Campus Id	`InArgument<Int32>`	No	Select a value in the drop-down list of the activity in the Designer window.

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
High Start Date	`InArgument<DateTime>`	Yes	The High Start Date captured from an event. **Note**: You can capture a range of dates by specifying different values in the High Start Date and Low Start Date fields. If you are not checking for a range of dates, use the same value in the High Start Date and Low Start Date fields.
Low Start Date	`InArgument<DateTime>`	Yes	The Low Start Date captured from an event.
Terms List	`OutArgument<TermEntity[]>`	Yes	The Term List returned by the lookup function. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities** , select **TermEntity**, and click **OK**.  Name — Variable type NewTermList — TermEntity[] See TermEntity Class in the Anthology Student Object Library.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

SaveStudentCourse (V2)

The SaveStudentCourse activity enables you to Register or Unregister a Student Course. You can also transfer students who have been registered for a course from one class section to another class section using the TransferClassSection action in the SaveStudentCourse activity.

SaveStudentCourse is used after a CreateStudentCourse (V2) activity has created a Student Course instance. SaveStudentCourse will persist a Student Course instance in the database.



CreateStudentCourse

Student Id: StudId

Student Enrollment Period Id: StudEnroll.Id

Class Section Id: ClassSects(0).Id

SaveStudentCourse

Action: Register

Properties

Cmc.Nexus.Academics.Workflow.SaveStudentCourse

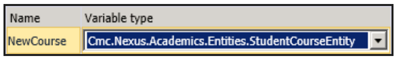
Misc

Action	Cmc.Nexus.Academics.Workflow.CourseAction.Register
DisplayName	SaveStudentCourse
ParentTermId	513
StudentCourse	NewCourse
StudentCourseId	*Enter a VB expression*
TransferToClassSectionId	*Enter a VB expression*
ValidationMessages	*Enter a VB expression*

Properties

SaveStudentCourse Properties

Property	Value	Required	Notes
Action	`InArgument <CourseAction>`	Yes	Select one of the following options: - Register - Unregister - TransferClassSection When the action TransferClassSection is selected, the StudentCourseId and TransferToClassSectionId are required.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
ParentTermId	`InArgument<Int32>`	No	Use this value when a Parent/Child relationship has been defined for the terms at your institution and you want to register a student into a Child term. The ParentTermId value is the AdTermId of the Parent term in Anthology Student. You can use LookupTerms (V2) to pass the Id into the SaveStudentCourse activity. **Note**: This value is used only with the Register Action when registering a student into a child term.

Property	Value	Required	Notes
StudentCourse	`InArgument <StudentCourseEntity>`	Conditional	The Student Course created by this workflow activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. **Note**: This value is used only with the Register and Unregister Actions. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Academics.Contracts > Cmc.Nexus.Academics.Entities**, select **StudentCourseEntity**, and click **OK**.  Name Variable type NewCourse Cmc.Nexus.Academics.Entities.StudentCourseEntity See StudentCourseEntity Class in the Anthology Student Object Library.
StudentCourseId	`InArgument<Int32>`	Conditional	The StudentCourseId is the StudentCourse.Id (which is AdEnrollSched.AdEnrollSchedID in Anthology Student for the current class). This value is used only with the TransferClassSection Action.
TransferToClassSectionId	`InArgument<Int32>`	Conditional	The TransferToClassSectionId is the ClassSection.Id of the class into which you want to transfer students (mapped to AdClassSched.AdClassSchedId in Anthology Student). **Note**: This value is used only with the TransferClassSection Action.

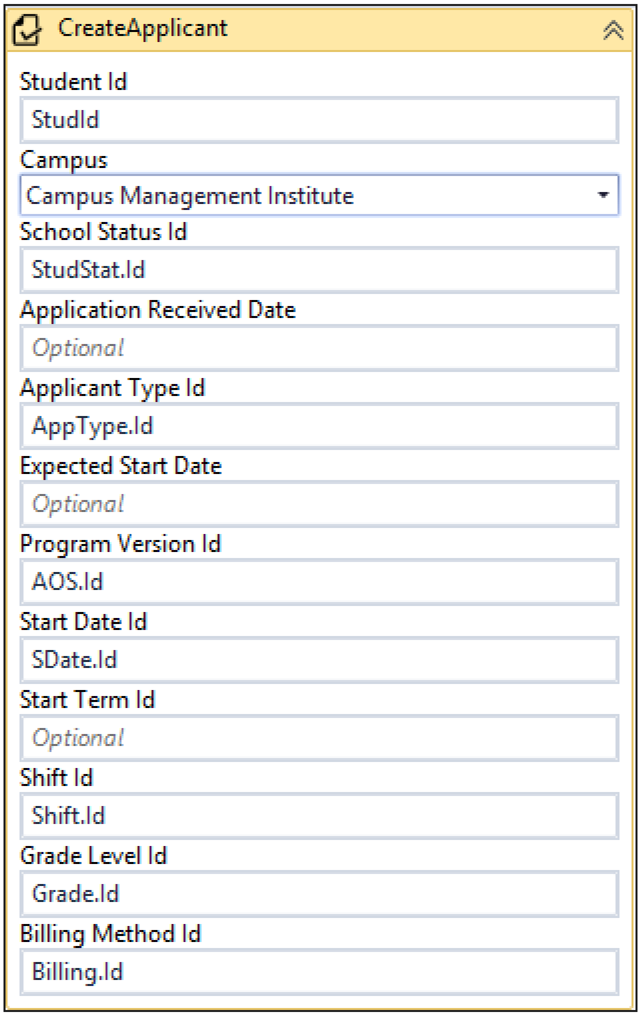
Property	Value	Required	Notes
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

Cmc.Nexus.Admissions.Workflow

CreateApplicant

You can use the CreateApplicant activity to dynamically create Applicant records in Anthology Student based on the data retrieved from an online form.

The CreateApplicant activity creates an instance of an Applicant record; it does not save the record to the database. The workflow can include other activities that manipulate the record before it is saved. To persist the Applicant record in the database, insert a SaveApplicant activity.



CreateApplicant

Student Id
StudId

Campus
Campus Management Institute

School Status Id
StudStat.Id

Application Received Date
Optional

Applicant Type Id
AppType.Id

Expected Start Date
Optional

Program Version Id
AOS.Id

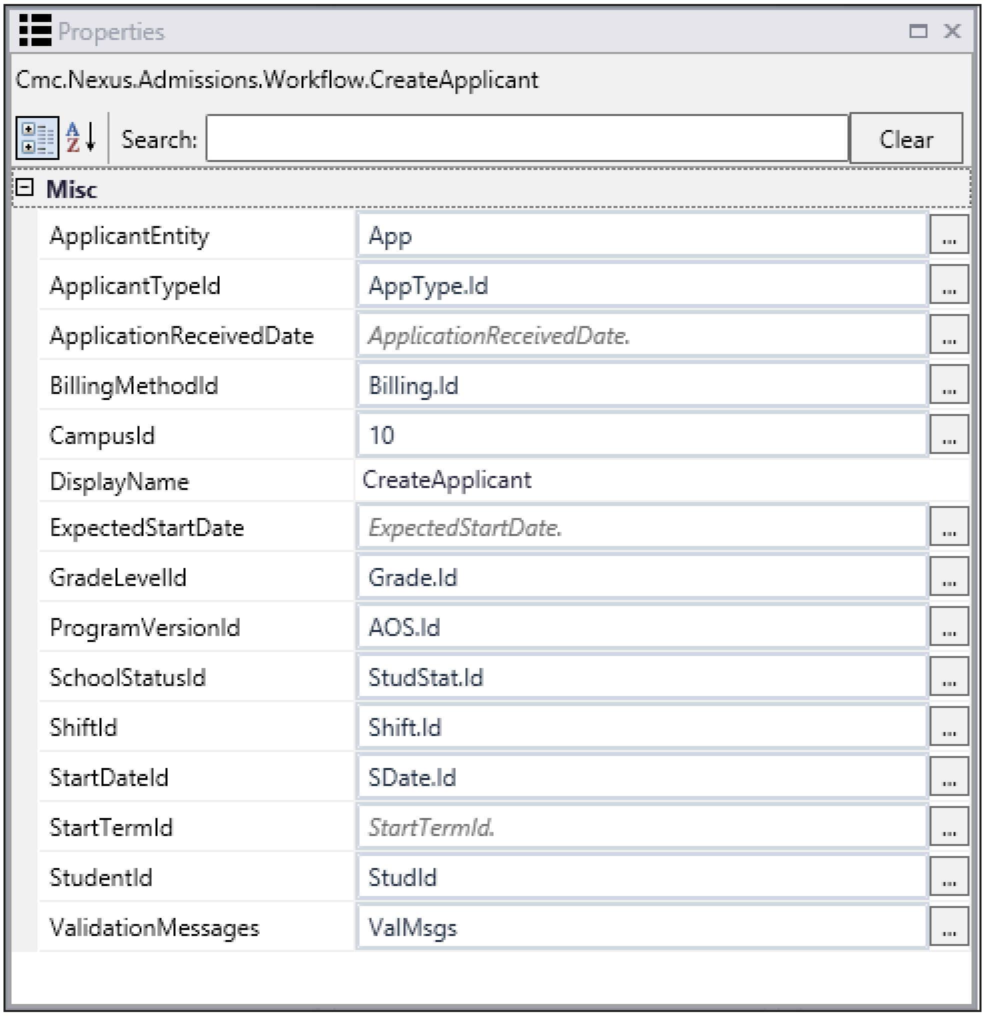
Start Date Id
SDate.Id

Start Term Id
Optional

Shift Id
Shift.Id

Grade Level Id
Grade.Id

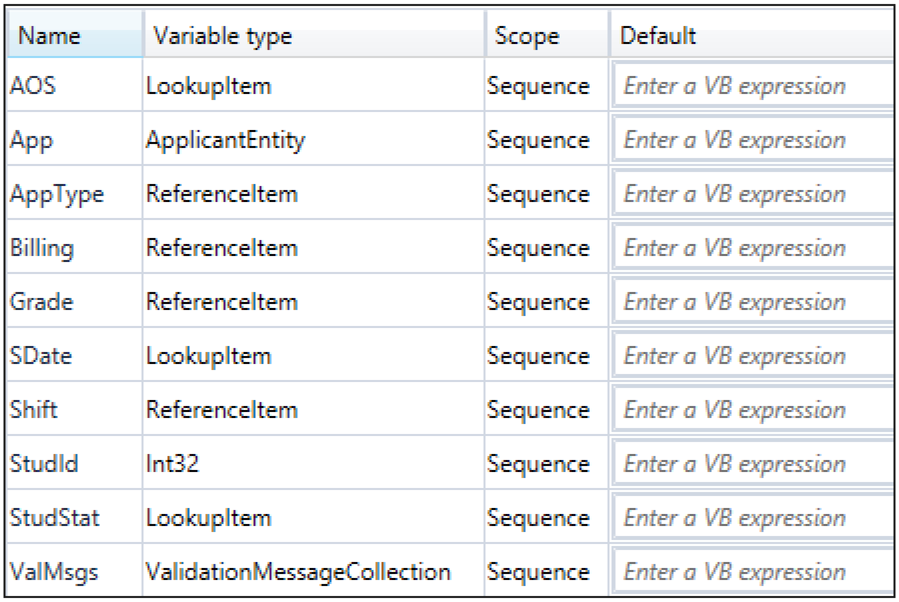
Billing Method Id
Billing.Id



Cmc.Nexus.Admissions.Workflow.CreateApplicant

ApplicantEntity	App
ApplicantTypeId	AppType.Id
ApplicationReceivedDate	*ApplicationReceivedDate.*
BillingMethodId	Billing.Id
CampusId	10
DisplayName	CreateApplicant
ExpectedStartDate	*ExpectedStartDate.*
GradeLevelId	Grade.Id
ProgramVersionId	AOS.Id
SchoolStatusId	StudStat.Id
ShiftId	Shift.Id
StartDateId	SDate.Id
StartTermId	*StartTermId.*
StudentId	StudId
ValidationMessages	ValMsgs

The following variable definitions are used in the CreateApplicant example above.

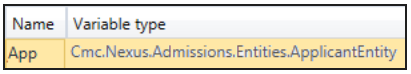


Name	Variable type	Scope	Default
AOS	LookupItem	Sequence	*Enter a VB expression*
App	ApplicantEntity	Sequence	*Enter a VB expression*
AppType	ReferenceItem	Sequence	*Enter a VB expression*
Billing	ReferenceItem	Sequence	*Enter a VB expression*
Grade	ReferenceItem	Sequence	*Enter a VB expression*
SDate	LookupItem	Sequence	*Enter a VB expression*
Shift	ReferenceItem	Sequence	*Enter a VB expression*
StudId	Int32	Sequence	*Enter a VB expression*
StudStat	LookupItem	Sequence	*Enter a VB expression*
ValMsgs	ValidationMessageCollection	Sequence	*Enter a VB expression*

The variables are populated by using lookup activities preceding the CreateApplicant activity in the workflow.

Properties

CreateApplicant Properties

Property	Value	Required	Notes
ApplicantEntity	`OutArgument <ApplicantEntity>`	Yes	The Applicant created by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types....** In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Admissions.Contracts > Cmc.Nexus.Admissions.Entities**, and select **ApplicantEntity**.  Name · Variable type App · Cmc.Nexus.Admissions.Entities.ApplicantEntity See ApplicantEntity Class in the Anthology Student Object Library.
ApplicantTypeId	`InArgument<Int32>`	No	Specify the Applicant Type Id using a VB expression or variable.
ApplicationReceivedDate	`InArgument<DateTime>`	No	Specify the application received date using a VB expression or variable.
BillingMethodId	`InArgument<Int32>`	No	Specify the Billing Method Id using a VB expression or variable.
CampusId	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
ExpectedStartDate	`InArgument<DateTime>`	No	Specify the expected start date using a VB expression or variable
GradeLevelId	`InArgument<Int32>`	No	Specify the database identifier for the Grade Level using a VB expression or variable.

Property	Value	Required	Notes
ProgramVersionId	`InArgument<Int32>`	No	Specify the Area of Study Id using a VB expression or variable.
SchoolStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable
ShiftId	`InArgument<DateTime>`	No	Specify the identifier for the shift using a VB expression or variable.
StartDateId	`InArgument<DateTime>`	No	Specify the identifier for the student's start date using a VB expression or variable
StartTermId	`InArgument<DateTime>`	No	Specify the identifier for the student's start term using a VB expression or variable.
StudentId	`InArgument<Int32>`	Yes	Specify the Student Id using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

CreatePortalAccount

The CreatePortalAccount activity automates the creation of AD and Portal accounts based on the triggering event. For example, a Portal account can be created when a specific status change occurs or when a new applicant/lead completes a form.

The supported authentication methods include STS, AD, and Azure AD.

The StudentEntity Saved Event is the preferred event to call the CreatePortalAccount activity especially for AD and Azure AD authentication.

Use Case

An institution uses Anthology Student and implements a workflow with CreatePortalAccount activity to create a Portal account when a New Lead is created. A common scenario is that the activity is triggered by a StudentEntity Saved Event (web client) or Person Saved Event (desktop client), however, the workflow could also be triggered by a different event.

Properties

CreatePortalAccount Properties

Property	Value	Required	Notes
AddUserToActiveDirectory	`InArgument<Boolean>`	Yes if AD or Azure AD is used	A Boolean expression that specifies whether the user needs to be added to the Active Directory. The default value is False. Set this value to True if Active Directory (AD) or Azure AD is used in your Portal. **Prerequisite**: If the Portal is deployed in an AD environment, a username and password for a "Student Active Directory User" must be configured in the Portal Admin Console . The CreatePortalAccount activity uses the "Student Active Directory User" account as an impersonation account to call the Create/Update WebPortalAccountService APIs.
AdGuId	`InArgument<Guid>`	Yes if AD or Azure AD is used	Specify the globally unique identifier (GUID) (stored in wpUser.GUID of the Portal database) using a VB expression or variable.

Property	Value	Required	Notes
CampusId	`InArgument<Int32>`	Yes	Specify the database identifier for the student's Campus using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Email	`InArgument<String>`	No	Specify the student's email address using a VB expression or variable.
FirstName	`InArgument<String>`	No	Specify the student's first name using a VB expression or variable.
LastName	`InArgument<String>`	No	Specify the student's last name using a VB expression or variable.
NewId	`OutArgument<Int32>`	No	Specify the new Id using a VB expression or variable. This value will be used if the activity is used to update a Student Portal account.
Password	`InArgument<String>`	Yes	Specify a value for the initial password using a VB expression or variable. **Note**: The initial password must comply with the given password rules. The CreatePortalAccount activity will fail if the password is not strong enough and doesn't follow all rules, especially in Azure AD environments. An uppercase letter, lowercase letter, number, and symbol may all have to be used. Even when the strong password requirement disabled in Anthology Student (least restrictive), AzureAD may still block risky passwords.
StudentId	`InArgument<Int32>`	Yes	Specify a student identifier (i.e., syStudentId from the syStudent table) using a VB expression or variable.

Property	Value	Required	Notes
UserCode	`InArgument<String>`	Yes	Specify a unique user code (stored in wpUser.UserCode of the Portal database) using a VB expression or variable. This will be the student's login Id for the Student Portal. **Note**: In Azure AD environments a domain name may need to be specified, e.g.: entity.FirstName + "." + entity.LastName + "@<*server*>.campusnexus.cloud"
ValidationMessages	`InOutArgument <ValidationMessage Collection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

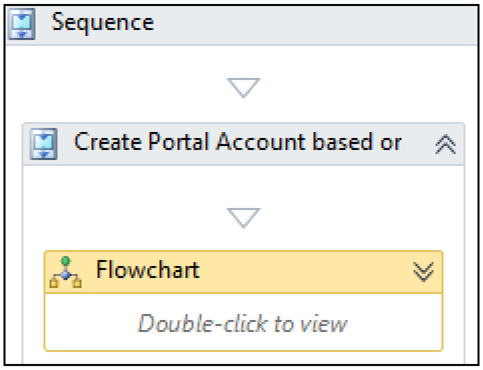
Example: Create Portal Account from a StudentEntity Saved Event in AD Environment

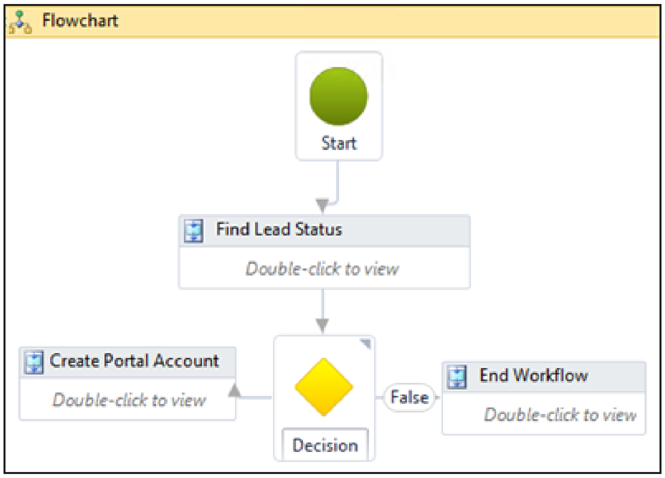
This is an example of Anthology Student eventing workflow for a StudentEntity Saved event in an AD environment. With a few minor changes to the example, a Person Saved Event can be used.

- If you are using the **Web Client** for Anthology Student, select the **StudentEntity Saved Event** when creating the workflow.
- If you are using the **Desktop Client** for Anthology Student, select the **Person Saved Event** when creating the workflow.

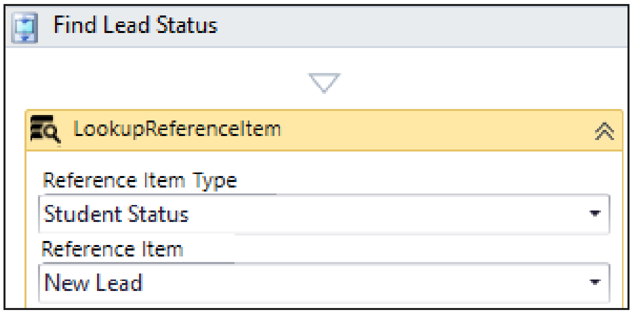
The workflow runs when a new lead or new student is added. It creates a username as "first.last" with password "nexus123$".

Note: If you want to create a Portal account based on a Forms Builder sequence being completed, you will need to create the form sequence and supporting workflow that will perform the status change or create the New Lead record. The status change or creation of a New Lead record will be the trigger for a separate workflow that will then create the Portal and/or AD account. For example, if a New Lead Record is created via a Forms Builder sequence, a separate workflow using the StudentEntity Saved Event would then trigger and create the Portal and/or AD account.





- The workflow is organized in a sequence named "Create Portal Account based on Lead" that contains a Flowchart.
- The Flowchart has a Start node, Decision node, and sequences named:
    - "Find Lead Status"
    - "Create Portal Account"
    - "End Workflow"
- "Find Lead Status" contains a LookupRefernceItem activity that checks for the "New Lead" status.



Reference Item Type
Student Status

Reference Item
New Lead

- The Decision evaluates a condition statement to true/-false:

  To check for a new student, specify:

  ```
  Entity.EntityState =
  Cmc.Core.EntityModel.EntityState.Added
  and Entity.SchoolStatusId = Lead.Id
  ```

- The "False" branch leads to the "End Workflow" sequence with a TerminateWorkflow activity.

Create Portal Account Sequence

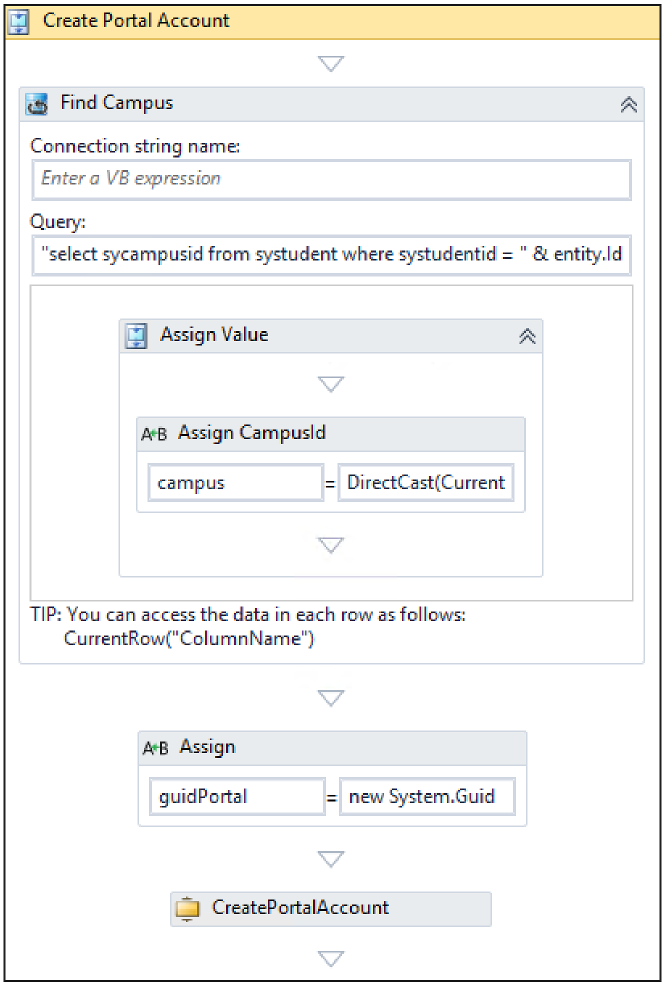
- The "True" branch leads to the "Create Portal Account" sequence with the following activities:
- ExecuteDataReader

  The ExecuteDataReader named "Find Campus" finds the `sycampusid`. The Query CommandText in the Query section is as follows:

  ```
  "select sycampusid from systudent
  where systudentid = " & entity.Id
  ```

  The Assign activity in the Query section assigns the `sycampusid` found in the database to the "campus" variable.

  ```
  DirectCast(CurrentRow("sycampusid"),
  ```



Create Portal Account

Find Campus

Connection string name:

Enter a VB expression

Query:

"select sycampusid from systudent where systudentid = " & entity.Id

Assign Value

Assign CampusId

campus = DirectCast(Current

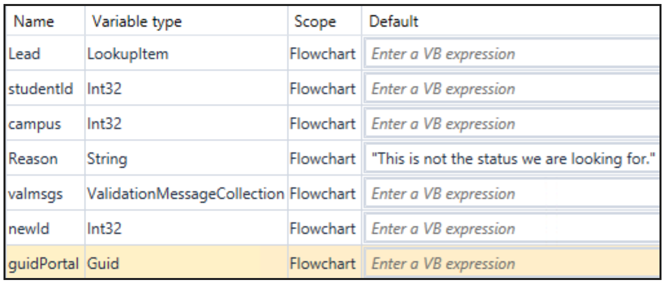
TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

Assign

guidPortal = new System.Guid

CreatePortalAccount

Variables:



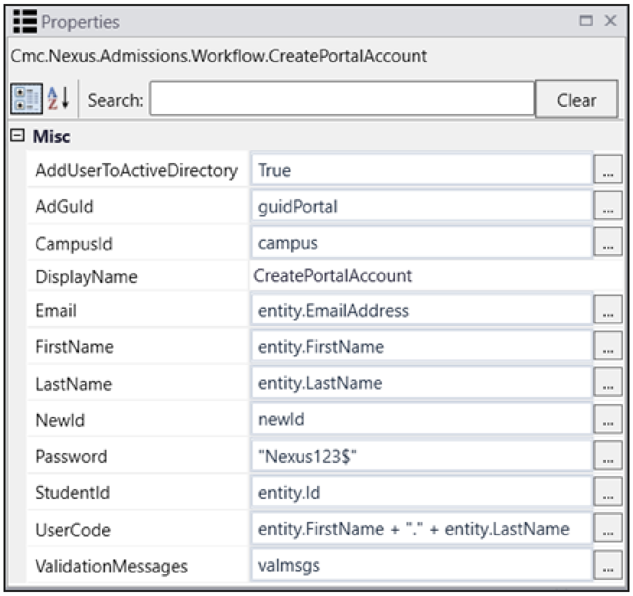
Name	Variable type	Scope	Default
Lead	LookupItem	Flowchart	Enter a VB expression
studentId	Int32	Flowchart	Enter a VB expression
campus	Int32	Flowchart	Enter a VB expression
Reason	String	Flowchart	"This is not the status we are looking for."
valmsgs	ValidationMessageCollection	Flowchart	Enter a VB expression
newId	Int32	Flowchart	Enter a VB expression
guidPortal	Guid	Flowchart	Enter a VB expression

int32)

- Assign

  The Assign activity below ExecuteDataReader assigns the value "new System.Guid" to the "guidPortal" variable.

- CreatePortalAccount



Properties

Cmc.Nexus.Admissions.Workflow.CreatePortalAccount

Search: Clear

Misc

AddUserToActiveDirectory	True
AdGuId	guidPortal
CampusId	campus
DisplayName	CreatePortalAccount
Email	entity.EmailAddress
FirstName	entity.FirstName
LastName	entity.LastName
NewId	newId
Password	"Nexus123$"
StudentId	entity.Id
UserCode	entity.FirstName + "." + entity.LastName
ValidationMessages	valmsgs

Note: In an Azure AD environment make sure that the CreatePortalAccount activity in the "StudentEntity Saved" workflow has a fully qualified name in the UserCode property, e.g., first.last@*<server>*customer.campusnexus.cloud,

Usage in AD and Azure AD Environments with Forms Builder

In addition to specific Properties for the CreatePortalAccount in AD and Azure AD environments, please note the following requirements/limitations:

AD Environments with Forms Builder

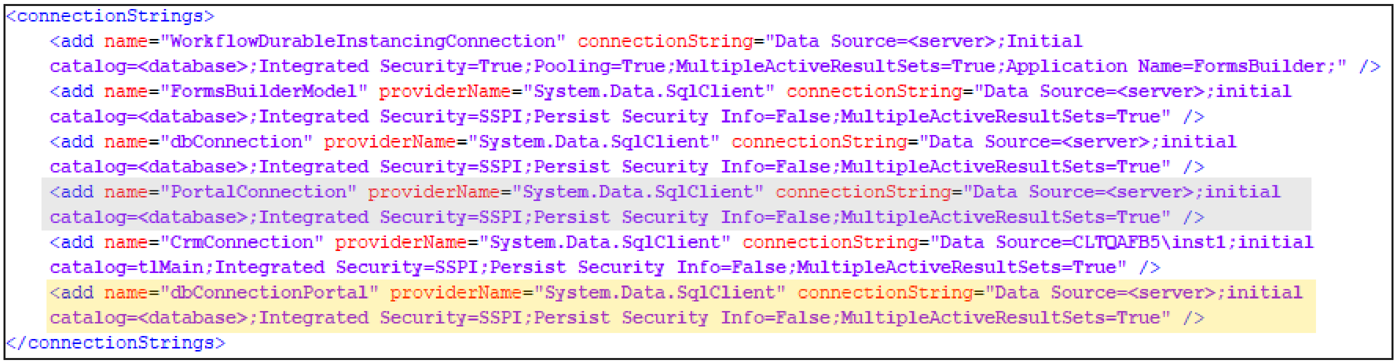
In AD environments, the CreatePortalAccount activity within a Forms Builder workflow (i.e., not as directed in the separate StudentEntity Saved Event workflow) will function only if a 2nd Portal connection string is added to the Renderer web.config file.

The original connection string in the Renderer web.config is:

```
<add name="PortalConnection" providerName="System.Data.SqlClient" con-
nectionString="Data Source=...
```

The added connection string for the CreatePortalAccount activity is:

```
<add name="dbConnectionPortal" providerName="System.Data.SqlClient" con-
nectionString="Data Source=...
```



```
<connectionStrings>
    <add name="WorkflowDurableInstancingConnection" connectionString="Data Source=<server>;Initial
    catalog=<database>;Integrated Security=True;Pooling=True;MultipleActiveResultSets=True;Application Name=FormsBuilder;" />
    <add name="FormsBuilderModel" providerName="System.Data.SqlClient" connectionString="Data Source=<server>;initial
    catalog=<database>;Integrated Security=SSPI;Persist Security Info=False;MultipleActiveResultSets=True" />
    <add name="dbConnection" providerName="System.Data.SqlClient" connectionString="Data Source=<server>;initial
    catalog=<database>;Integrated Security=SSPI;Persist Security Info=False;MultipleActiveResultSets=True" />
    <add name="PortalConnection" providerName="System.Data.SqlClient" connectionString="Data Source=<server>;initial
    catalog=<database>;Integrated Security=SSPI;Persist Security Info=False;MultipleActiveResultSets=True" />
    <add name="CrmConnection" providerName="System.Data.SqlClient" connectionString="Data Source=CLTQAFB5\inst1;initial
    catalog=tlMain;Integrated Security=SSPI;Persist Security Info=False;MultipleActiveResultSets=True" />
    <add name="dbConnectionPortal" providerName="System.Data.SqlClient" connectionString="Data Source=<server>;initial
    catalog=<database>;Integrated Security=SSPI;Persist Security Info=False;MultipleActiveResultSets=True" />
</connectionStrings>
```

Azure AD Environments

When you use Forms Builder to create a New Lead in Anthology Student and you want to create a Portal account and/or AD account, you must create a separate workflow using the **StudentEntity Saved Event**.

a. Forms Builder will create the New Lead.

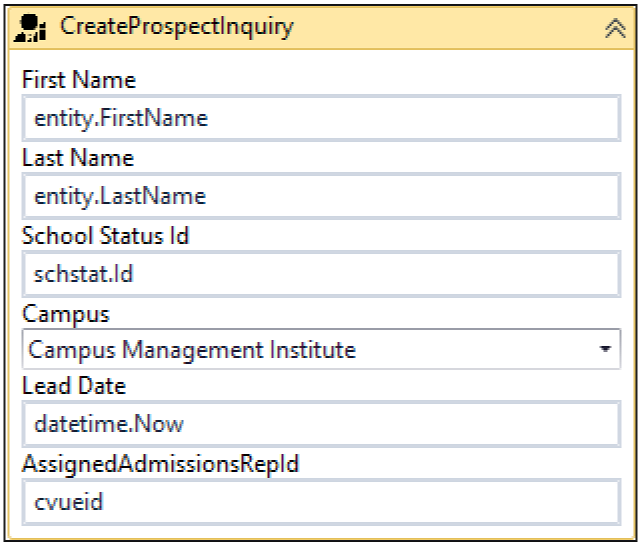
b. "StudentEntity Saved" workflow logic will trigger and create the accounts.

It is best practice, if you want to trigger the account creation based on status change, to always have a separate workflow when Forms Builder is involved to avoid duplicates.

CreateProspectInquiry

You can use the CreateProspectInquiry activity to dynamically create an instance of a ProspectInquiryEntity record based on the data retrieved from an online Request for Information (RFI) form.

The CreateProspectInquiry activity does not save the record to the database. The workflow can include other activities that manipulate the record before it is saved. To persist the record in the database, use a SaveProspectInquiry activity.



CreateProspectInquiry

First Name: entity.FirstName

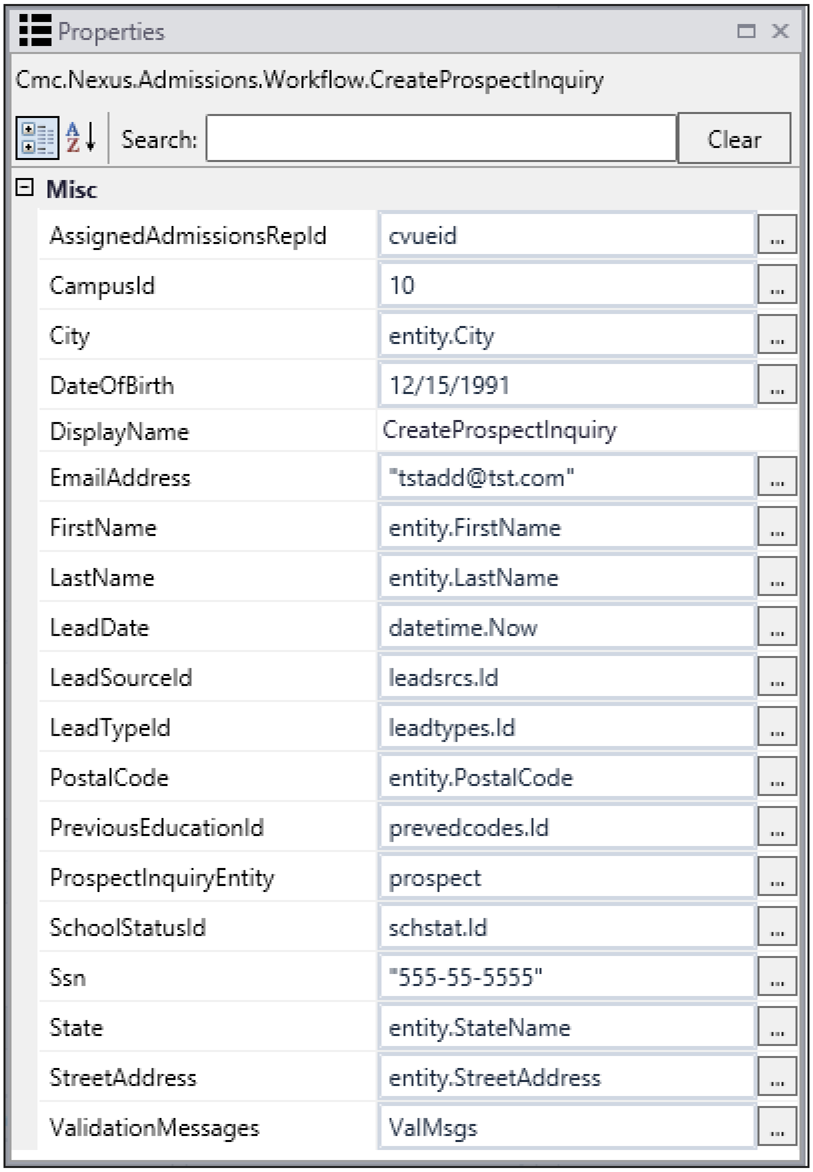
Last Name: entity.LastName

School Status Id: schstat.Id

Campus: Campus Management Institute

Lead Date: datetime.Now

AssignedAdmissionsRepId: cvueid



Cmc.Nexus.Admissions.Workflow.CreateProspectInquiry

Misc

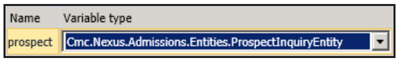
AssignedAdmissionsRepId	cvueid
CampusId	10
City	entity.City
DateOfBirth	12/15/1991
DisplayName	CreateProspectInquiry
EmailAddress	"tstadd@tst.com"
FirstName	entity.FirstName
LastName	entity.LastName
LeadDate	datetime.Now
LeadSourceId	leadsrcs.Id
LeadTypeId	leadtypes.Id
PostalCode	entity.PostalCode
PreviousEducationId	prevedcodes.Id
ProspectInquiryEntity	prospect
SchoolStatusId	schstat.Id
Ssn	"555-55-5555"
State	entity.StateName
StreetAddress	entity.StreetAddress
ValidationMessages	ValMsgs

Properties

CreateProspectInquiry Properties

Property	Value	Required	Notes
AssignedAdmissionsRepId	`InArgument<Int32>`	Yes	Specify the assigned Admissions Representative Type Entity Id using a VB expression or variable.
CampusId	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.

Property	Value	Required	Notes
City	`InArgument<String>`	Conditional	Specify name of the city in the student's address, if address information is provided, using a VB expression or variable.
DateOfBirth	`InArgument<DateTime>`	Conditional	Specify the student's date of birth using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EmailAddress	`InArgument<String>`	Conditional	Specify the student's email address, if provided, using a VB expression or variable.
FirstName	`InArgument<String>`	Yes	Specify the student's first name using a VB expression or variable.
LastName	`InArgument<String>`	Yes	Specify the student's last name using a VB expression or variable.
LeadDate	`InArgument<DateTime>`	Yes	Specify the lead date using a VB expression or variable.
LeadSourceId	`InArgument<Int32>`	Conditional	Specify the lead source identifier, if provided, using a VB expression or variable.
LeadTypeId	`InArgument<Int32>`	Conditional	Specify the lead type identifier, if provided, using a VB expression or variable.
PostalCode	`InArgument<String>`	Conditional	Specify the student's postal code, if provided, using a VB expression or variable.
PreviousEducationId	`InArgument<Int32>`	Conditional	Specify the student's previous education identifier, e.g., high school, if provided, using a VB expression or variable.

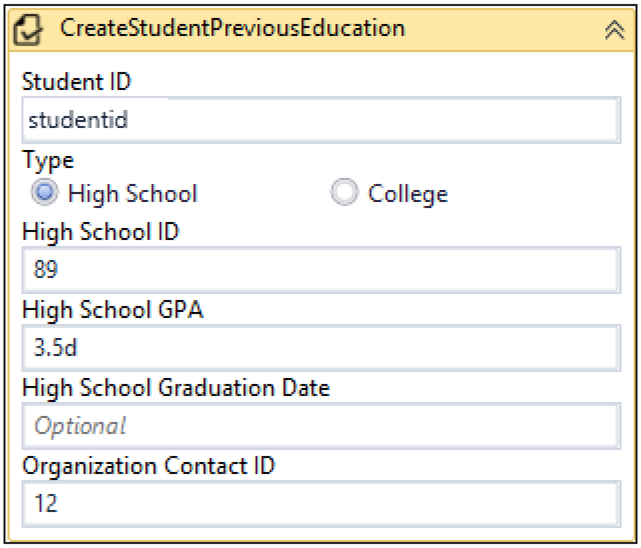
Property	Value	Required	Notes
ProspectInquiryEntity	`OutArgument <ProspectInquiryEntity>`	Yes	The Prospect Inquiry Entity created by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Admissions.Contracts > Cmc.Nexus.Admissions.Entities**, select **ProspectInquiryEntity** and click **OK**.  Name: prospect Variable type: Cmc.Nexus.Admissions.Entities.ProspectInquiryEntity See ProspectInquiryEntity Class in the Anthology Student Object Library.
SchoolStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
Ssn	`InArgument<String>`	Conditional	Specify the student's social security number, if provided, using a VB expression or variable.
State	`InArgument<String>`	Conditional	Specify name of the state in the student's address, if address information is provided, using a VB expression or variable.
StreetAddress	`InArgument<String>`	Conditional	Specify the student's street address, if address information is provided, using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

CreateStudentPreviousEducation

You can use the CreateStudentPreviousEducation activity to dynamically create an instance of a record in the amProspectPrevEduc table. The previous education data can be high school or college information. The data can be retrieved from an online application form or directly inserted in the activity (and its properties).

The CreateStudentPreviousEducation activity does not save the record to the database. The workflow can include other activities that manipulate the record before it is saved. To persist the record in the database, use a SaveStudentPreviousEducation activity.

The example below shows the activity and properties for the selection Type = High School.



CreateStudentPreviousEducation

Student ID

studentid

Type

● High School ○ College

High School ID

89

High School GPA

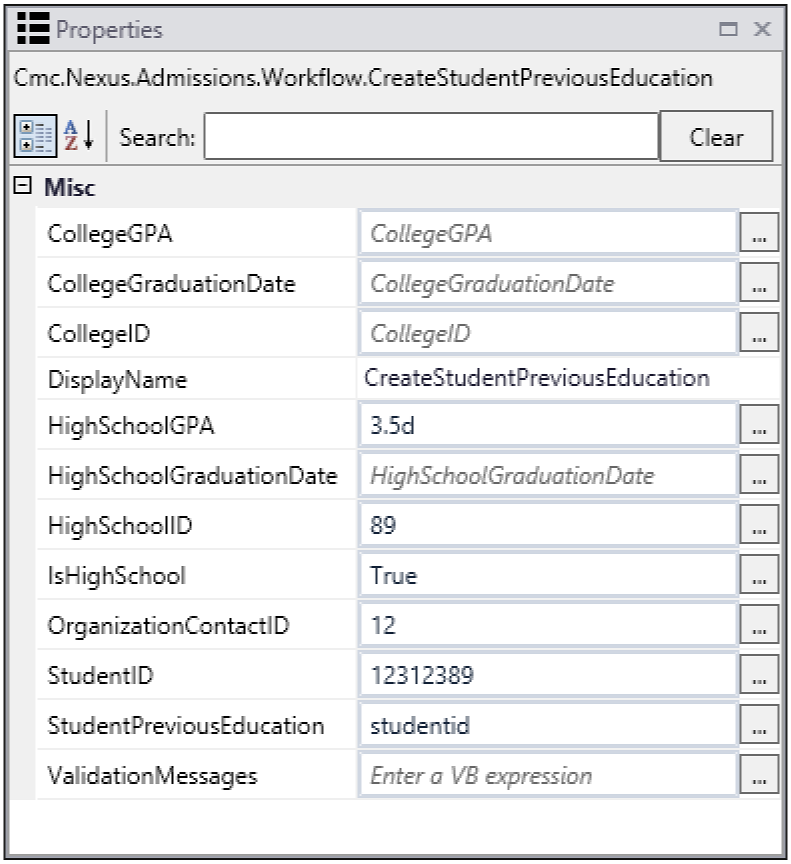
3.5d

High School Graduation Date

Optional

Organization Contact ID

12

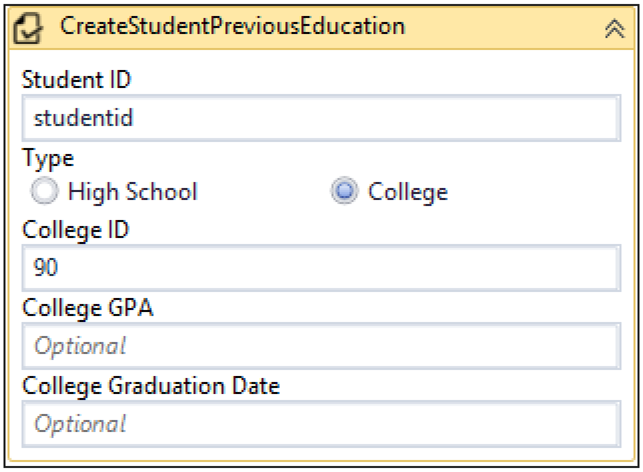


Cmc.Nexus.Admissions.Workflow.CreateStudentPreviousEducation

⊟ **Misc**

CollegeGPA	*CollegeGPA*
CollegeGraduationDate	*CollegeGraduationDate*
CollegeID	*CollegeID*
DisplayName	CreateStudentPreviousEducation
HighSchoolGPA	3.5d
HighSchoolGraduationDate	*HighSchoolGraduationDate*
HighSchoolID	89
IsHighSchool	True
OrganizationContactID	12
StudentID	12312389
StudentPreviousEducation	studentid
ValidationMessages	*Enter a VB expression*

The example below shows the activity and properties for the selection Type=College.



CreateStudentPreviousEducation

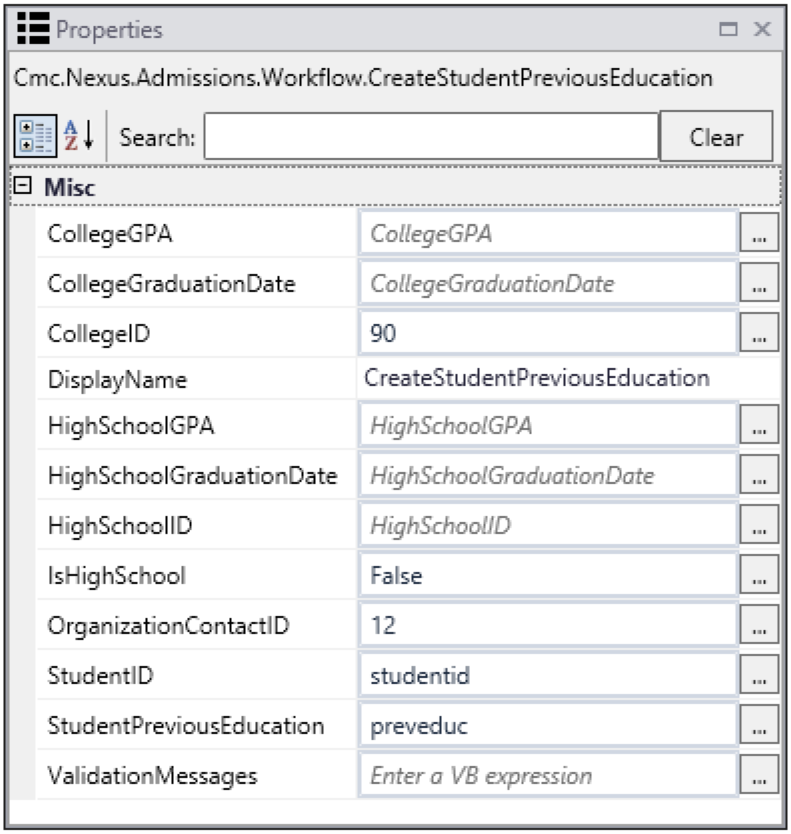
Student ID: studentid

Type: ○ High School ● College

College ID: 90

College GPA: *Optional*

College Graduation Date: *Optional*



Cmc.Nexus.Admissions.Workflow.CreateStudentPreviousEducation

Misc

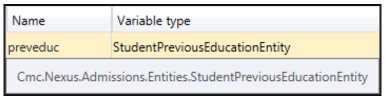
CollegeGPA	*CollegeGPA*
CollegeGraduationDate	*CollegeGraduationDate*
CollegeID	90
DisplayName	CreateStudentPreviousEducation
HighSchoolGPA	*HighSchoolGPA*
HighSchoolGraduationDate	*HighSchoolGraduationDate*
HighSchoolID	*HighSchoolID*
IsHighSchool	False
OrganizationContactID	12
StudentID	studentid
StudentPreviousEducation	preveduc
ValidationMessages	*Enter a VB expression*

Properties

CreateStudentPreviousEducation Properties

Property	Value	Required	Notes
CollegeGPA	`InArgument<Decimal>`	No	Specify the student's College GPA, if provided, using a VB expression or variable, for example `4.0d`.
CollegeGraduationDate	`InArgument<DateTime>`	No	Specify the student's College Graduation Date, if provided, using a VB expression or variable.
CollegeId	`InArgument<Int32>`	Conditional	Specify the College Identifier, if provided, using a VB expression or variable. The College Id is required if the selection for previous education Type = College; it is optional for Type = High School.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

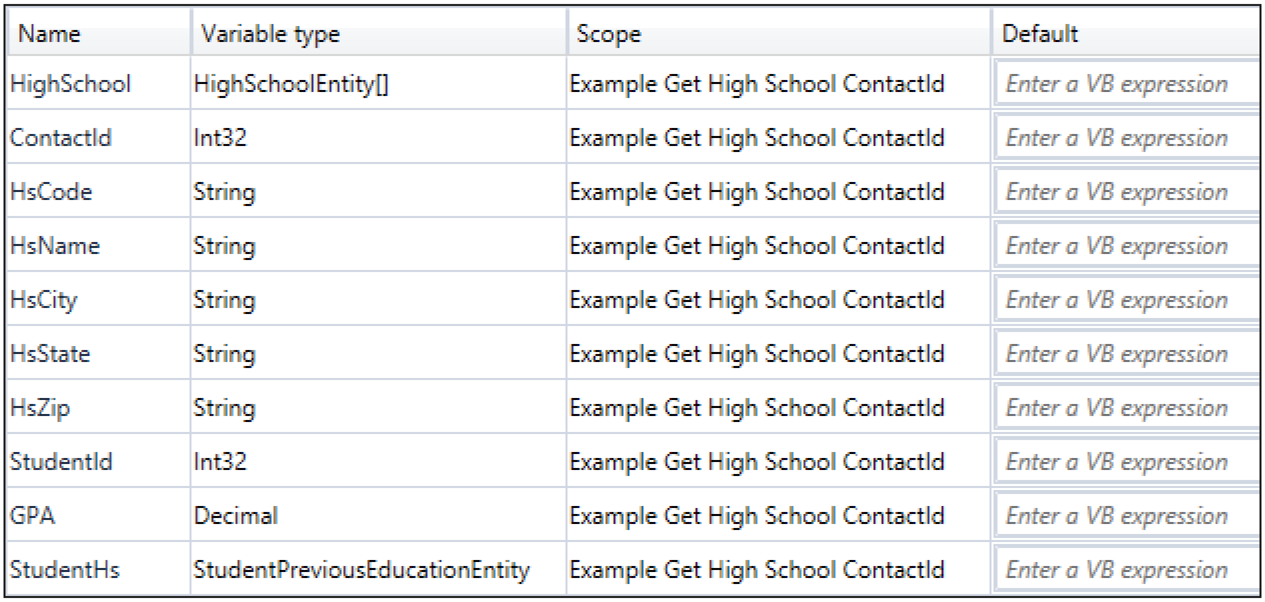
Property	Value	Required	Notes
HighSchoolGPA	`InArgument<Decimal>`	Conditional	Specify the student's High School GPA using a VB expression or variable, for example `3.5d`. The High School GPA is required if the selection for previous education Type = High School; it is optional for Type = College.
HighSchoolGraduationDate	`InArgument<DateTime>`	No	Specify the student's High School Graduation Date, if provided, using a VB expression or variable.
HighSchoolId	`InArgument<String>`	Conditional	Specify the High School Identifier using a VB expression or variable. The High School Id is required if the selection for previous education Type = High School; it is optional for Type = College.
IsHighSchool	`InArgument<Boolean>`	Yes	A Boolean expression that specifies whether the selection for previous education Type = High School (default) or College.
OrganizationContactId	`InArgument<String>`	No	Specify the Organization Contact Identifier using a VB expression or variable. The OrganizationContactId is not required when creating the previous education entity; however, if your institution wants to include this in the workflow, refer to the workflow sequence below. This sequence gives you an example of how to look up a high school, get the contact id for that organization, and pass it to the CreateStudentPreviousEducation activity.
StudentId	`InArgument<String>`	Yes	Specify a Student Id using a VB expression or variable.

Property	Value	Required	Notes
StudentPreviousEducation	`OutArgument <StudentPreviousEducationEntity>`	Yes	The Student Previous Education Entity created by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Admissions.Contracts > Cmc.Nexus.Admissions.Entities**, and select **StudentPreviousEducationEntity**.  Name Variable type preveduc StudentPreviousEducationEntity Cmc.Nexus.Admissions.Entities.StudentPreviousEducationEntity See StudentPreviousEducationEntity Class in the Anthology Student Object Library.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

Get OrganizationContactId Sequence

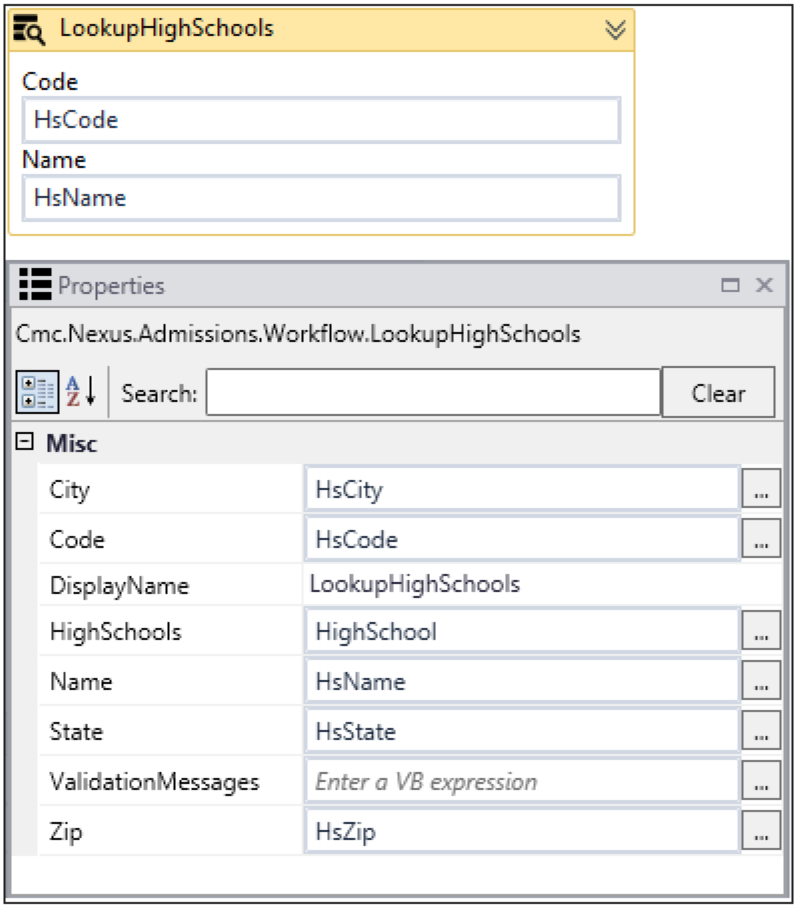
The following workflow sequence provides a query to obtain the OrganizationContactId for a high school.

The sequence uses the following variables:



Name	Variable type	Scope	Default
HighSchool	HighSchoolEntity[]	Example Get High School ContactId	*Enter a VB expression*
ContactId	Int32	Example Get High School ContactId	*Enter a VB expression*
HsCode	String	Example Get High School ContactId	*Enter a VB expression*
HsName	String	Example Get High School ContactId	*Enter a VB expression*
HsCity	String	Example Get High School ContactId	*Enter a VB expression*
HsState	String	Example Get High School ContactId	*Enter a VB expression*
HsZip	String	Example Get High School ContactId	*Enter a VB expression*
StudentId	Int32	Example Get High School ContactId	*Enter a VB expression*
GPA	Decimal	Example Get High School ContactId	*Enter a VB expression*
StudentHs	StudentPreviousEducationEntity	Example Get High School ContactId	*Enter a VB expression*

1. Use a LookupHighSchools activity.



LookupHighSchools

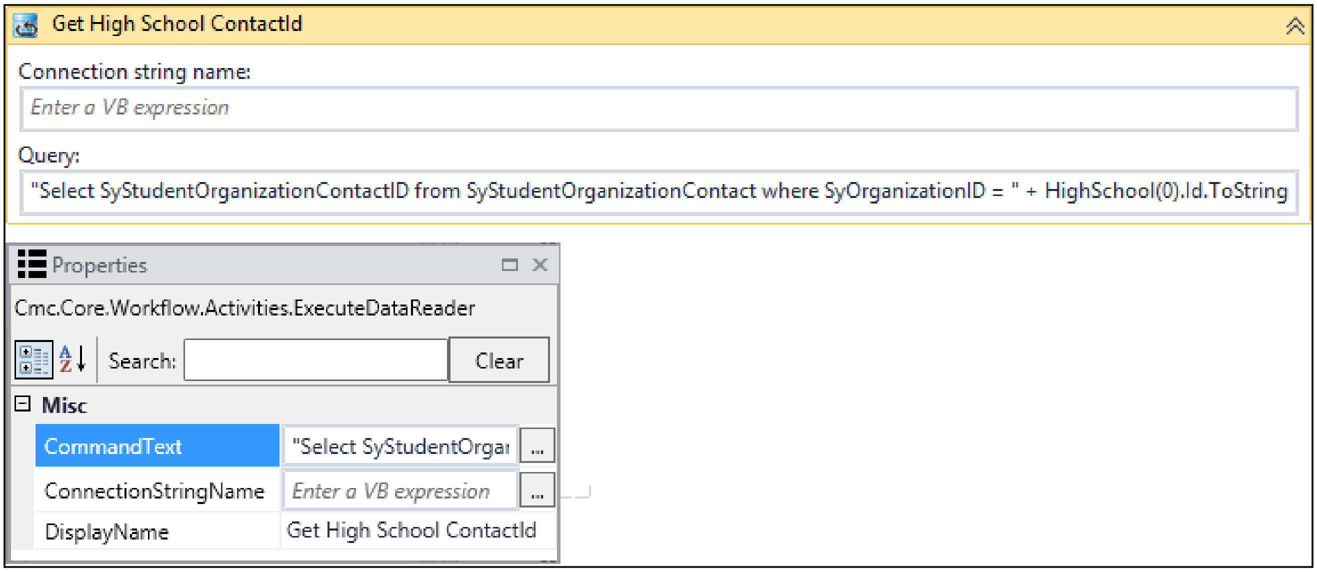
Code: HsCode

Name: HsName

Cmc.Nexus.Admissions.Workflow.LookupHighSchools

City	HsCity
Code	HsCode
DisplayName	LookupHighSchools
HighSchools	HighSchool
Name	HsName
State	HsState
ValidationMessages	*Enter a VB expression*
Zip	HsZip

2. Add an ExecuteDataReader activity to the sequence. Specify a CommandText (String) expression as shown below.



Get High School ContactId

Connection string name: *Enter a VB expression*

Query:

"Select SyStudentOrganizationContactID from SyStudentOrganizationContact where SyOrganizationID = " + HighSchool(0).Id.ToString

Properties

Cmc.Core.Workflow.Activities.ExecuteDataReader

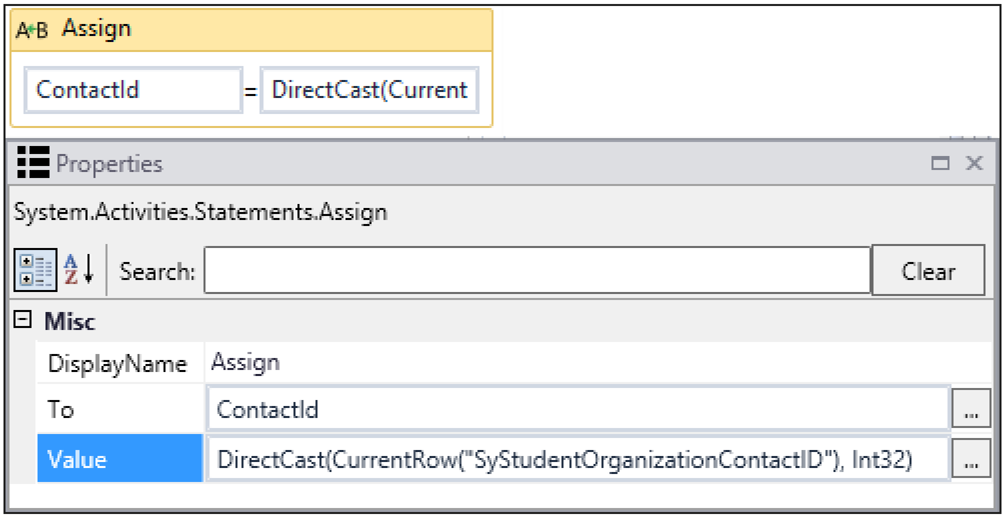
Misc

CommandText: "Select SyStudentOrgai

ConnectionStringName: *Enter a VB expression*

DisplayName: Get High School ContactId

3. Drag an Assign activity into the ExecuteDateReader activity. Associate the ContactId value with the SyStudentOrganizationContactId that was retrieved by the ExecuteDateReader activity.



A+B Assign

ContactId = DirectCast(Current

Properties

System.Activities.Statements.Assign

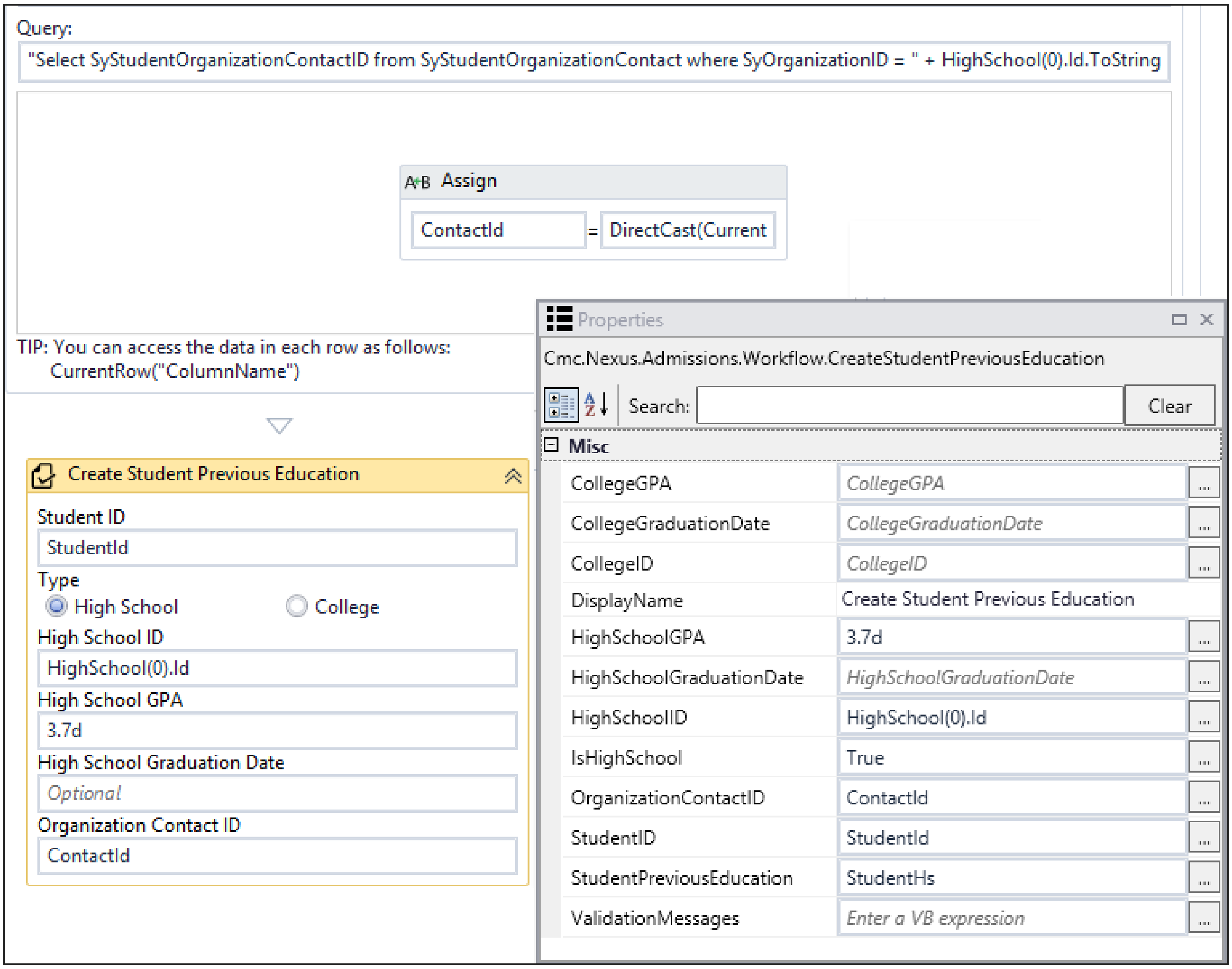
Misc

DisplayName: Assign

To: ContactId

Value: DirectCast(CurrentRow("SyStudentOrganizationContactID"), Int32)

4. Add a CreateStudentPreviousEducation activity to the sequence. Associate the OrganizationContactId with the ContactId from the Assign activity.



Query:

"Select SyStudentOrganizationContactID from SyStudentOrganizationContact where SyOrganizationID = " + HighSchool(0).Id.ToString

ContactId = DirectCast(Current

TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

Student ID: StudentId

Type: ● High School ○ College

High School ID: HighSchool(0).Id

High School GPA: 3.7d

High School Graduation Date: *Optional*

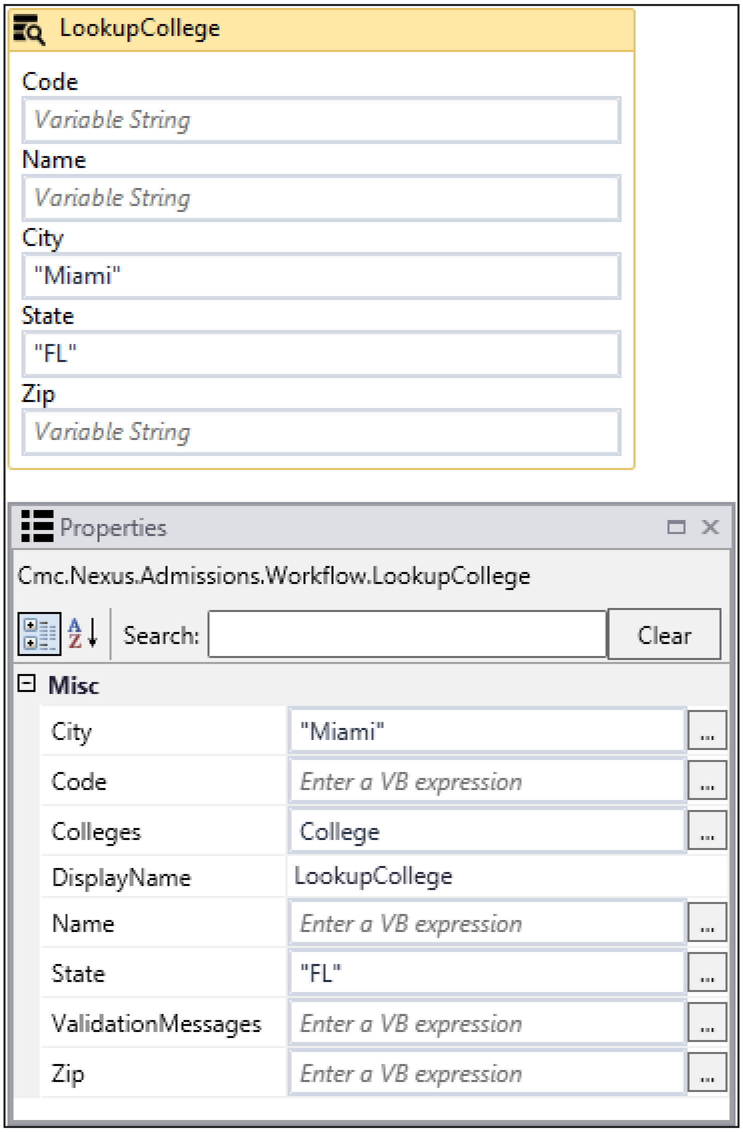
Organization Contact ID: ContactId

Cmc.Nexus.Admissions.Workflow.CreateStudentPreviousEducation

CollegeGPA	*CollegeGPA*
CollegeGraduationDate	*CollegeGraduationDate*
CollegeID	*CollegeID*
DisplayName	Create Student Previous Education
HighSchoolGPA	3.7d
HighSchoolGraduationDate	*HighSchoolGraduationDate*
HighSchoolID	HighSchool(0).Id
IsHighSchool	True
OrganizationContactID	ContactId
StudentID	StudentId
StudentPreviousEducation	StudentHs
ValidationMessages	*Enter a VB expression*

LookupCollege

The LookupCollege activity returns an array of Colleges based on filter criteria. The values are retrieved from the amCollege table in the Anthology Student database. The filters (in arguments) include City, Code, Name, State, and ZIP. At least one of the in arguments is required (C1).


LookupCollege

Code: *Variable String*

Name: *Variable String*

City: "Miami"

State: "FL"

Zip: *Variable String*

Properties

Cmc.Nexus.Admissions.Workflow.LookupCollege

Search:

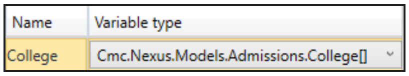
Misc

City	"Miami"
Code	*Enter a VB expression*
Colleges	College
DisplayName	LookupCollege
Name	*Enter a VB expression*
State	"FL"
ValidationMessages	*Enter a VB expression*
Zip	*Enter a VB expression*

Properties

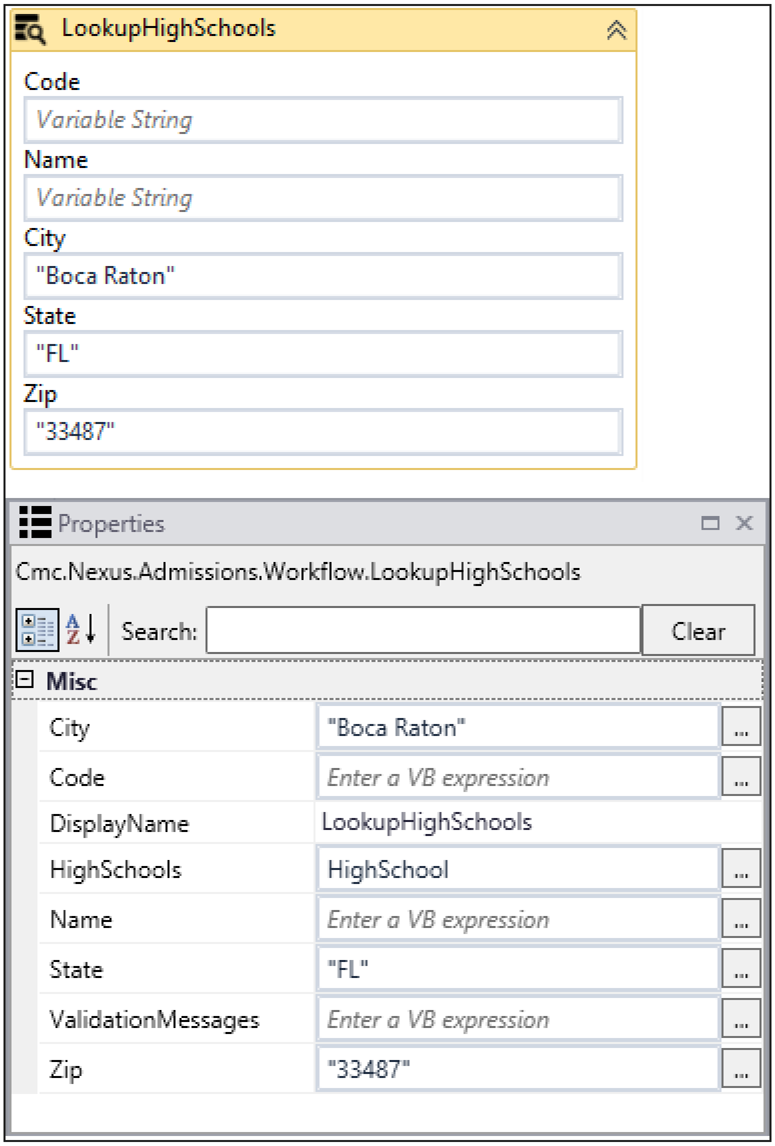
LookupCollege Properties

Property	Value	Required	Notes
City	`InArgument<String>`	C1	Specify name of the city of the college location using a VB expression or variable.

Property	Value	Required	Notes
Code	`InArgument<String>`	C1	Specify code of the college using a VB expression or variable.
Colleges	`OutArgument<College[]>`	Yes	The array of colleges retrieved by this workflow activity. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Models.Admissions**, select **College**, and click **OK**.  Name / Variable type College / Cmc.Nexus.Models.Admissions.College[] See CollegeEntity Class in the Anthology Student Object Library.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Name	`InArgument<String>`	C1	Specify the name of the college using a VB expression or variable.
State	`InArgument<String>`	C1	Specify name of the state of the college location using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.
Zip	`InArgument<String>`	C1	Specify the ZIP code of the college location using a VB expression or variable

LookupHighSchools

The LookupHighSchools activity returns an array of HighSchools based on filter criteria. The values are retrieved from the amHighSchool table in the Anthology Student database. The filters (in arguments) include City, Code, Name, State, and ZIP. At least one of the in arguments is required (C1).



LookupHighSchools

Code: *Variable String*

Name: *Variable String*

City: "Boca Raton"

State: "FL"

Zip: "33487"

Properties

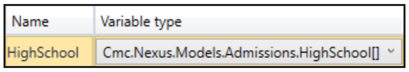
Cmc.Nexus.Admissions.Workflow.LookupHighSchools

City	"Boca Raton"
Code	*Enter a VB expression*
DisplayName	LookupHighSchools
HighSchools	HighSchool
Name	*Enter a VB expression*
State	"FL"
ValidationMessages	*Enter a VB expression*
Zip	"33487"

Properties

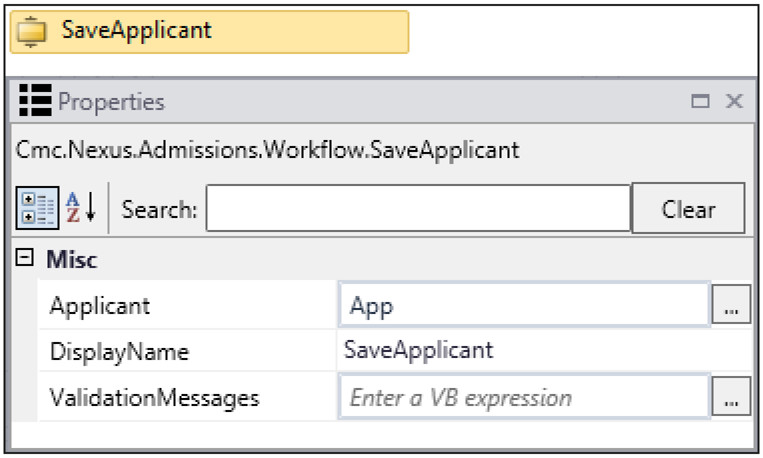
LookupHighSchools Properties

Property	Value	Required	Notes
City	`InArgument<String>`	C1	Specify name of the city of the high school location using a VB expression or variable.

Property	Value	Required	Notes
Code	`InArgument<String>`	C1	Specify code of the high school using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
HighSchools	`OutArgument<HighSchool[]>`	Yes	The array of high schools retrieved by this workflow activity. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Models.Admissions**, select **HighSchool**, and click **OK**.  Name | Variable type HighSchool | Cmc.Nexus.Models.Admissions.HighSchool[] See HighSchoolEntity Class in the Anthology Student Object Library.
Name	`InArgument<String>`	C1	Specify the name of the high school using a VB expression or variable.
State	`InArgument<String>`	C1	Specify name of the state of the high school location using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.
Zip	`InArgument<String>`	C1	Specify the ZIP code of the high school location using a VB expression or variable

SaveApplicant

The SaveApplicant activity saves an Applicant record that was created with the CreateApplicant activity.



SaveApplicant

Properties

Cmc.Nexus.Admissions.Workflow.SaveApplicant

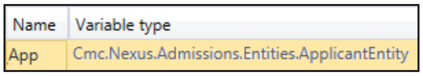
Search: Clear

Misc

Applicant	App
DisplayName	SaveApplicant
ValidationMessages	*Enter a VB expression*

Properties

SaveApplicant Properties

Property	Value	Required	Notes
Applicant	`InOutArgument<ApplicantEntity>`	Yes	Specify the Applicant entity to be saved using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Admissions.Contracts > Cmc.Nexus.Admissions.Entities**, and select **ApplicantEntity**.  Name | Variable type App | Cmc.Nexus.Admissions.Entities.ApplicantEntity See ApplicantEntity Class in the Anthology Student Object Library.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

SaveProspectInquiry

The SaveProspectInquiry activity saves a record that was created with the CreateProspectInquiry activity.

The result of the SaveProspectInquiry activity depends on whether the data passed in by the activity exists in the Anthology Student database and whether multiple inquiries are allowed in Anthology Student.

a. If the name and address data passed in the SaveProspectInquiry does not exist in Anthology Student, a SyStudent record (that includes a Prospect record) and SyStudentInquiry record will be created as an instance of a prospect.

b. If the name and address data passed in the SaveProspectInquiry exist in Anthology Student and multiple inquiries are allowed in Anthology Student, the existing SyStudent record is looked up and a new value is saved in the prospect collection (SyStudentInquiry).

c. If the name and address data passed in the SaveProspectInquiry exist in Anthology Student and multiple inquiries are not allowed in Anthology Student, the SyStudent record is updated with the values passed in by the activity.

To check the setting for multiple inquiries in Anthology Student, navigate to Setup > Campus Locations > select a campus > Add/Edit (button) > Allow... (tab). If "Track Multiple Lead Inquiries" is selected, the duplicate check function is enabled (see case b).

Note

The *Leads* web service provides the following configuration options in the <appSettings> section of the web.-config file:

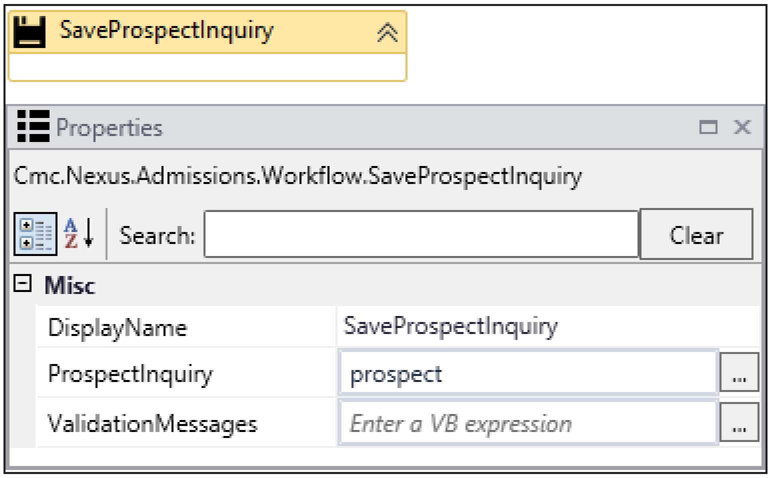
```
<add key="NewLeadSingleDuplicateHandling" value="I" />
```

— OR —

```
<add key="NewLeadSingleDuplicateHandling" value="E" />
```

Where value="I" indicates that prospect inquiry records are checked for duplicates and written to the SyStudent and SyStudentInquiry tables as described above (cases a, b, and c).

If the Leads API is configured with key value=E, duplicate prospect inquiry records are written to the electronic leads table (AmElectronicLeads). Regardless if a single or multiple duplicates are found, the prospect will always be processed and added to the AmElectronicLeads table. For more information, see "Duplicate Lead Validation, Configuration, and Interpreting the Response" in the Service Catalog.



SaveProspectInquiry

Properties

Cmc.Nexus.Admissions.Workflow.SaveProspectInquiry

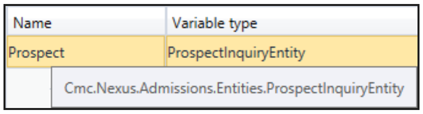
Search: Clear

Misc

DisplayName	SaveProspectInquiry
ProspectInquiry	prospect
ValidationMessages	*Enter a VB expression*

Properties

SaveProspectInquiry Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
ProspectInquiryEntity	`OutArgument <ProspectInquiryEntity>`	Yes	The Prospect Inquiry Entity created by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Admissions.Contracts > Cmc.Nexus.Admissions.Entities**, and select **ProspectInquiryEntity**.  Name | Variable type Prospect | ProspectInquiryEntity Cmc.Nexus.Admissions.Entities.ProspectInquiryEntity See ProspectInquiryEntity Class in the Anthology Student Object Library.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

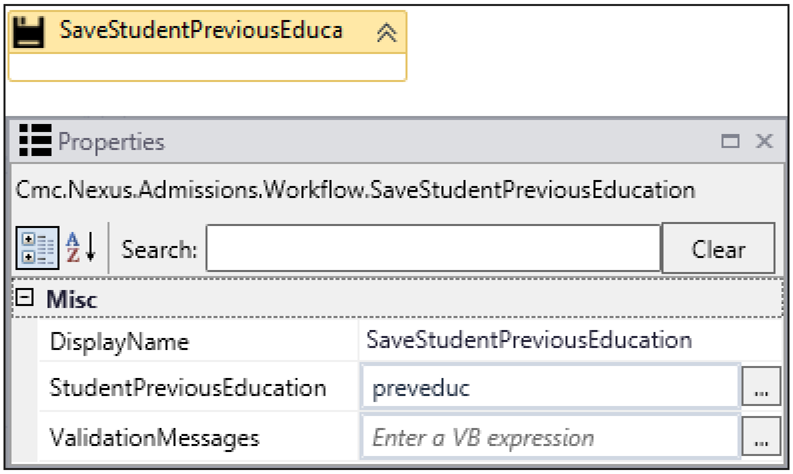
Database Fields

The SaveProspectInquiry activity can update the following fields in the database:

- Required fields:
  - StudentEntity.FirstName
  - StudentEntity.LastName
  - StudentEntity.SchoolStatusId
  - ProspectInquiryEntity.CampusId
  - ProspectInquiryEntity.LeadDate
  - ProspectInquiryEntity.AssignedAdmissionsRepId
- Optional fields:
  - ProspectInquiryEntity.LeadSourceId
  - ProspectInquiryEntity.LeadTypeId
  - StudentEntityEntity.DateOfBirth
  - StudentEntityEntity.Ssn
  - StudentEntityEntity.StreetAddress
  - StudentEntityEntity.PostalCode
  - StudentEntityEntity.EmailAddress
  - StudentEntityEntity.State
  - StudentEntityEntity.PreviousEducationId
  - StudentEntityEntity.WorkPhoneNumber
  - StudentEntityEntity.CitizenId
  - StudentEntityEntity.AlienNumber
  - StudentEntityEntity.City
  - StudentEntityEntity.CountyId
  - StudentEntityEntity.DriverLicenseNumber
  - StudentEntity.NationalityId
  - StudentEntity.NickName
  - StudentEntity.OtherEmailAddress
  - StudentEntity.OtherPhoneNumber

SaveStudentPreviousEducation

The SaveStudentPreviousEducation activity saves a record that was created with the CreateStudentPreviousEducation activity.



SaveStudentPreviousEduca

Properties

Cmc.Nexus.Admissions.Workflow.SaveStudentPreviousEducation

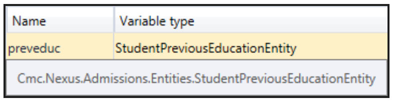
Search: Clear

Misc

DisplayName	SaveStudentPreviousEducation
StudentPreviousEducation	preveduc
ValidationMessages	*Enter a VB expression*

Properties

SaveStudentPreviousEducation Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
StudentPreviousEducation	`OutArgument <StudentPreviousEducationEntity>`	Yes	The Student Previous Education Entity created by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Admissions.Contracts > Cmc.Nexus.Admissions.Entities**, and select **StudentPreviousEducationEntity**.  Name: Variable type preveduc: StudentPreviousEducationEntity Cmc.Nexus.Admissions.Entities.StudentPreviousEducationEntity

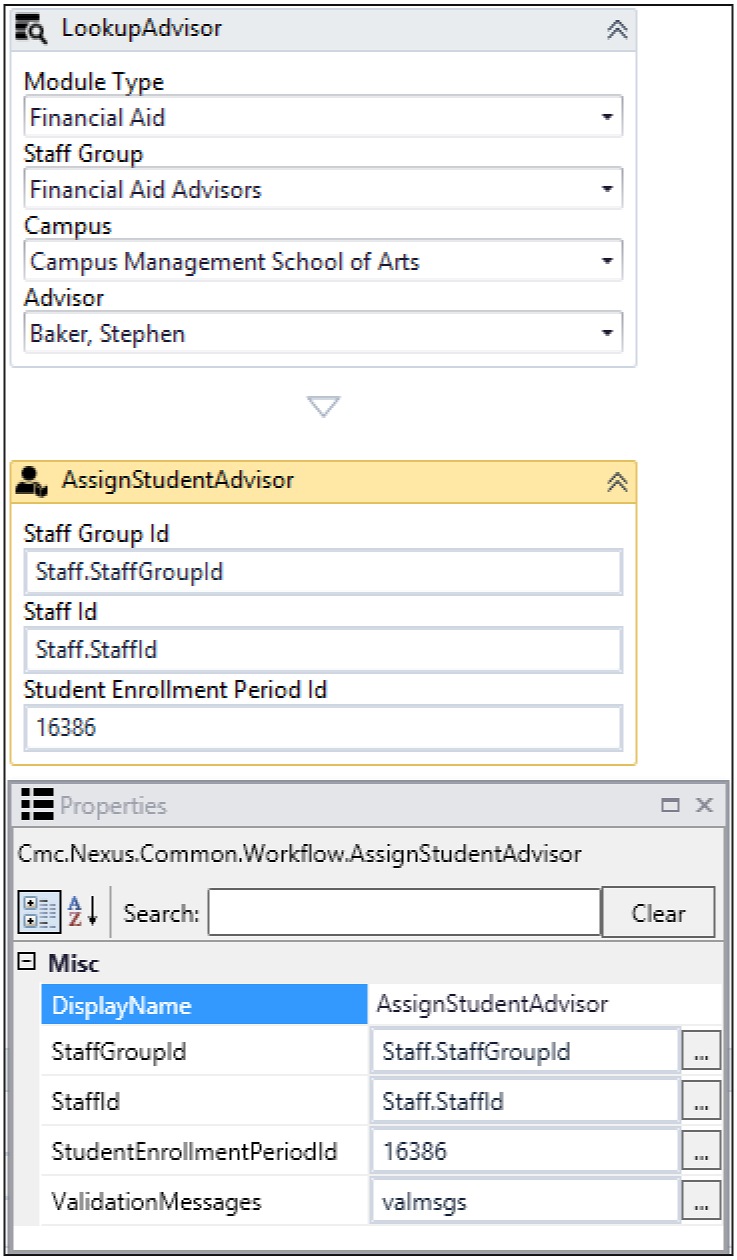
Property	Value	Required	Notes
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

Cmc.Nexus.Common.Workflow

AssignStudentAdvisor (V2)

The AssignStudentAdvisor (V2) activity enables you to assign a student advisor to a student. The activity derives the Advisor type (Academic = AD, Admissions = AM, etc.) based on the Staff Group selection. See StudentAdvisorEntityClass in the Anthology Student Object Library.

The AssignStudentAdvisor (V2) activity typically follows a LookupAdvisor (V2) activity, which uses the Module Type (e.g., Financial Aid), Staff Group (e.g., Financial Aid Advisors), Campus, and Advisor to determine the StaffGroupMember entity.



LookupAdvisor

Module Type: Financial Aid

Staff Group: Financial Aid Advisors

Campus: Campus Management School of Arts

Advisor: Baker, Stephen

AssignStudentAdvisor

Staff Group Id: Staff.StaffGroupId

Staff Id: Staff.StaffId

Student Enrollment Period Id: 16386

Properties

Cmc.Nexus.Common.Workflow.AssignStudentAdvisor

DisplayName	AssignStudentAdvisor
StaffGroupId	Staff.StaffGroupId
StaffId	Staff.StaffId
StudentEnrollmentPeriodId	16386
ValidationMessages	valmsgs

Properties

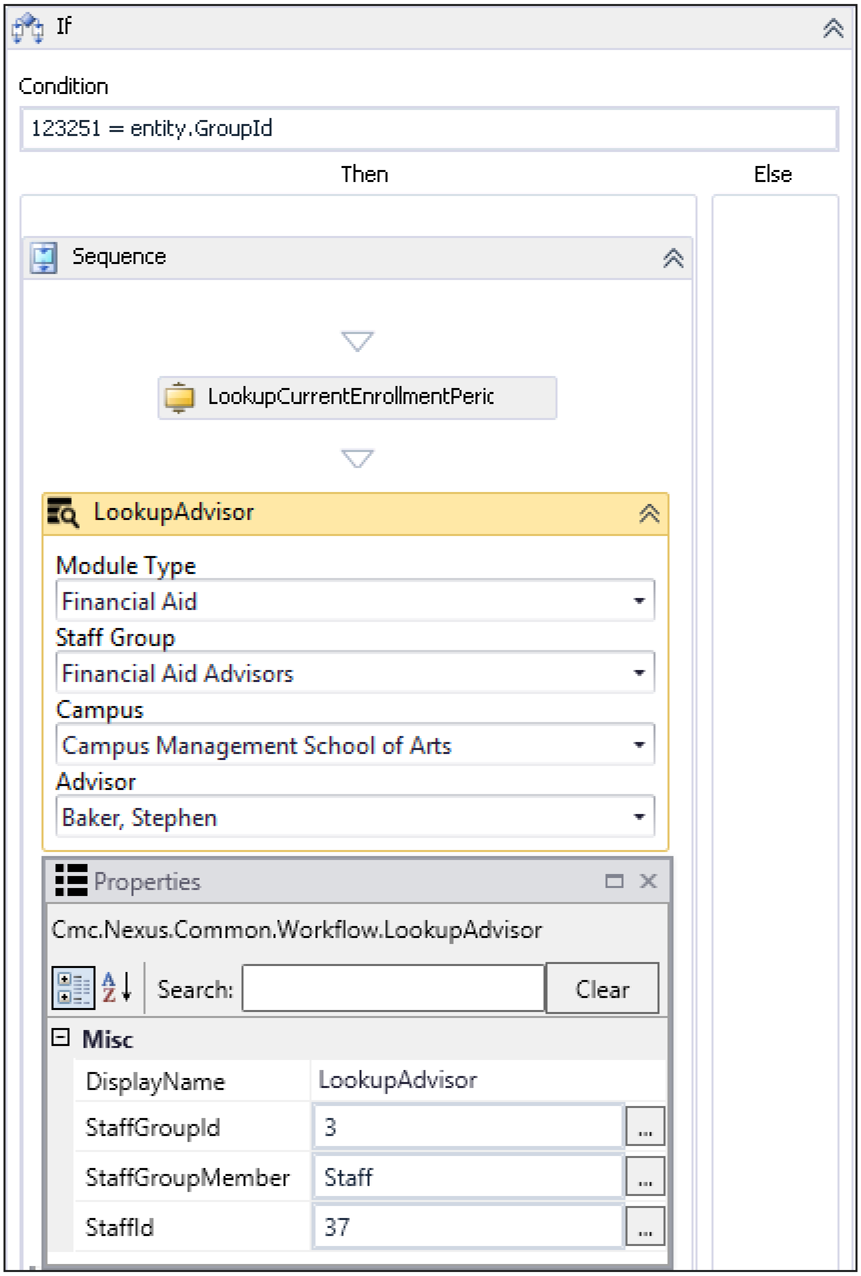
AssignStudentAdvisor Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
StaffGroupId	`InArgument<Int32>`	Yes	Specify the Staff Group Id using a VB expression or variable
StaffId	`InArgument<Int32>`	Yes	Specify the Staff Id using a VB expression or variable
StudentEnrollmentPeriodId	`InArgument<Int32>`	Yes	Specify the Student Enrollment Period Id using a VB expression or variable
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

LookupAdvisor (V2)

The LookupAdvisor (V2) activity looks up staff members that are flagged as an advisor in Anthology Student. The returned StaffGroupMember entity can be filtered by Staff Group, Campus, and Module Type (Academic = AD, Admissions = AM, etc.). The AssignStudentAdvisor (V2) activity can be used to assign the returned StaffGroupMember entity to a student.

If your institution assigns advisors as AD Advisor, an FA Advisor, a CS Advisor, etc. when a student enrolls, use the LookupStudentAdvisors (V2) activity instead of the LookupAdvisor (V2) activity.



Condition
123251 = entity.GroupId

Module Type: Financial Aid
Staff Group: Financial Aid Advisors
Campus: Campus Management School of Arts
Advisor: Baker, Stephen

Cmc.Nexus.Common.Workflow.LookupAdvisor

Search:

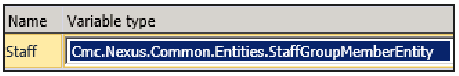
Misc

DisplayName	LookupAdvisor
StaffGroupId	3
StaffGroupMember	Staff
StaffId	37

Properties

LookupAdvisor Properties

Property	Value	Required	Notes
Advisor	`enum`	Yes	Select a value in the drop-down list of the activity in the Designer window. Advisors are filtered based on Campus. – OR – Specify the StaffId using a VB expression or variable.
Campus	`enum`	Yes	Select a value in the drop-down list of the activity in the Designer window.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Module Type	`enum`	Yes	Select a value in the drop-down list of the activity in the Designer window.
Staff Group	`enum`	Yes	Select a value in the drop-down list of the activity in the Designer window. Staff Groups are filtered based on Module Type selection. – OR – Specify the StaffGroupId using a VB expression or variable.
StaffGroupId	`InArgument<String>`	Yes	Specify the Staff Group Id using a VB expression or variable. – OR – Select a value in the drop-down list of the activity in the Designer window.

Property	Value	Required	Notes
StaffGroupMember	`OutArgument <StaffGroupMemberEntity>`	Yes	The LookupAdvisor activity returns the StaffGroupMember entity based on the selected Staff Group, Campus, and Module Type (Academic = AD, Admissions = AM, etc.) filter. The Id field (mapped to SyStaffByGroup.SyStaffByGroupId) is returned as "0". This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Common.Contracts > Cmc.Nexus.Common.Entities**, select **StaffGroupMemberEntity**, and click **OK**.  Name Variable type Staff Cmc.Nexus.Common.Entities.StaffGroupMemberEntity
StaffId	`InArgument<Int32>`	Yes	Specify the Staff Id using a VB expression or variable. – OR – Select a value in the drop-down list of the activity in the Designer window.

LookupReferenceItem

If you are using Workflow Composer with the Web Client for Anthology Student in an HTTPS environment, the LookupReferenceItem activity will fail unless you change the bindings in the WorkflowComposer.exe.config from HTTP to HTTPS. The WorkflowComposer.exe.config file is found in the C:\\Program Files (x86)\CMC\Workflow folder.

Edit the <bindings> as highlighted below:

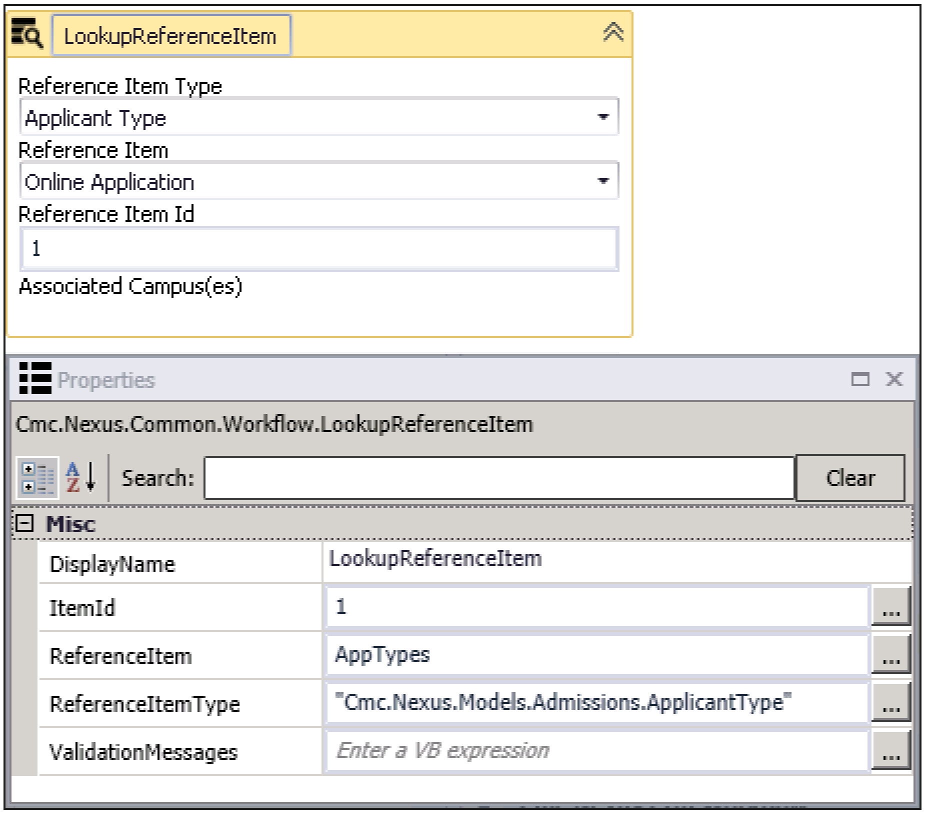
```
<bindings>
<!-- the binding named commonBinding will be applied to all service clients-->
<basicHttpsBinding>
<binding name="commonBinding" maxReceivedMessageSize="1073741824" closeTimeout="0:03:00"
receiveTimeout="00:3:00" sendTimeout="00:3:00"/>
</basicHttpsBinding>
</bindings>
```

The LookupReferenceItem activity can be used to retrieve a list of records from a selected Reference Item Type and allows you to select one of the records. This enables you to reference specific reference record data for use within a workflow. One common use for this is to populate the value of an attribute that is part of an entity record that will be created/updated within the workflow logic when an instance of the workflow is executed.

After you select the Reference Item Type from the drop-down list, the Reference Item drop-down list is populated with valid values for the selected Reference Item Type.

When a database contains multiple instances of the reference item type, e.g., in multiple campuses, a list of associated campus codes for each item is displayed in the **Associated Campus(es)** field so that the users can ensure that they are selecting the correct instance of the reference items for that workflow.

The example below shows Applicant Types as the Reference Item Type.

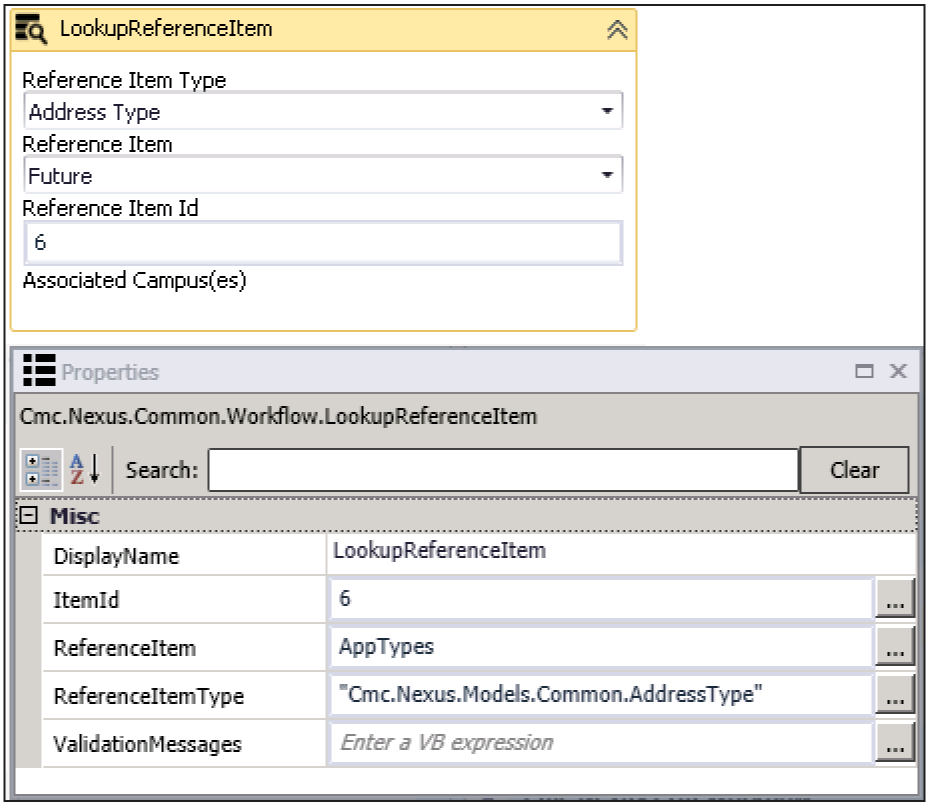

LookupReferenceItem

Reference Item Type
Applicant Type
Reference Item
Online Application
Reference Item Id
1
Associated Campus(es)

Cmc.Nexus.Common.Workflow.LookupReferenceItem

DisplayName	LookupReferenceItem
ItemId	1
ReferenceItem	AppTypes
ReferenceItemType	"Cmc.Nexus.Models.Admissions.ApplicantType"
ValidationMessages	*Enter a VB expression*

The example below shows Address Types as the Reference Item Type.


LookupReferenceItem

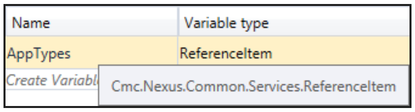
Reference Item Type
Address Type
Reference Item
Future
Reference Item Id
6
Associated Campus(es)

Cmc.Nexus.Common.Workflow.LookupReferenceItem

DisplayName	LookupReferenceItem
ItemId	6
ReferenceItem	AppTypes
ReferenceItemType	"Cmc.Nexus.Models.Common.AddressType"
ValidationMessages	*Enter a VB expression*

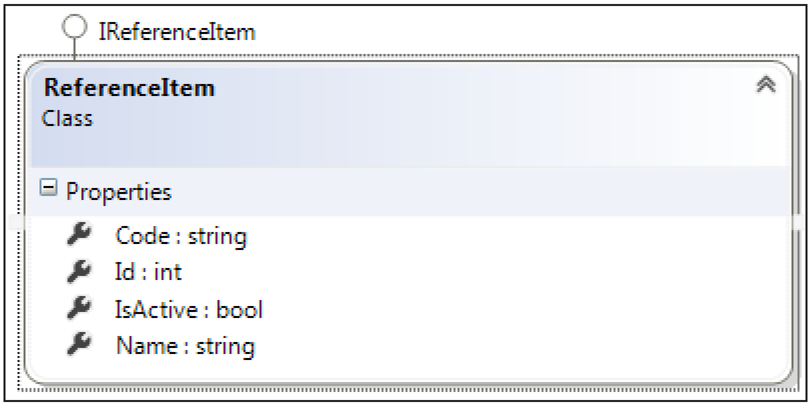
Properties

LookupReferenceItem Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
ItemId	`InArgument<Int32>`	Yes	The Item Id of the Reference Item selected to be looked up.
ReferenceItem	`OutArgument<ReferenceItem>`	Yes	The Reference Item returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Common.Contracts > Cmc.Nexus.Common.Services** , select **ReferenceItem**, and click **OK**.  Name / Variable type AppTypes / ReferenceItem Create Variabl Cmc.Nexus.Common.Services.ReferenceItem
ReferenceItemType	`InArgument<String>`	No	The Reference Item Type captured from an event. Select a value in the drop-down list of the activity in the Designer window. See Cmc.Nexus.Models for the mapping of entities.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

The properties of the ReferenceItem class are mapped to the following entities:

- `Code - <TableName>.Code`
- `Id - <Tablename>.<Tablename>ID example: AmApplicantTypeID`
- `IsActive - <Tablename>.Active`
- `Name - <Tablename>.Descrip`

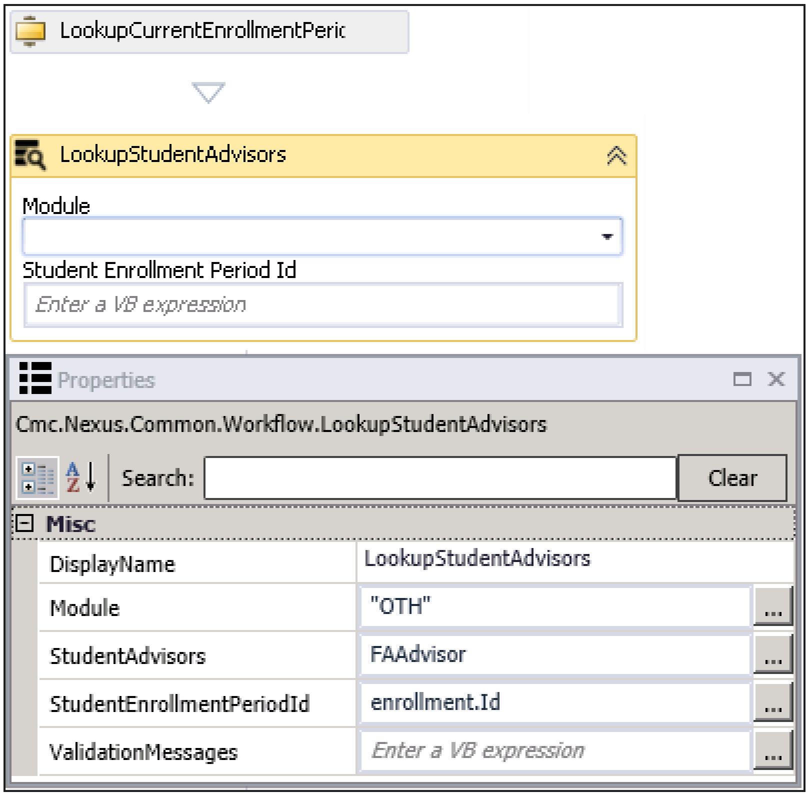

IReferenceItem
ReferenceItem
Class
Properties
Code : string
Id : int
IsActive : bool
Name : string

LookupStudentAdvisors (V2)

The LookupStudentAdvisors activity captures the Student Enrollment Period Id from an event and returns the Advisors that are currently assigned to a student. In most situations, the EnrollmentPeriod will only be associated with a single Advisor type. The Advisor type (Academic = AD, Admissions = AM, etc.) is selected in Module field of the lookup activity.

Based on the EnrollmentPeriod and Module, the LookupStudentAdvisors activity returns the staff member's FirstName, LastName, Module, and GroupName.

A use case for this activity is to assign a document or task to a staff member to follow up with a student when a specific event occurs, for example, the student is put on academic probation.



Module

Student Enrollment Period Id: Enter a VB expression

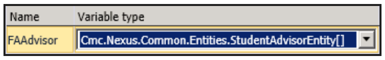
Cmc.Nexus.Common.Workflow.LookupStudentAdvisors

DisplayName	LookupStudentAdvisors
Module	"OTH"
StudentAdvisors	FAAdvisor
StudentEnrollmentPeriodId	enrollment.Id
ValidationMessages	Enter a VB expression

Properties

LookupStudentAdvisors Properties

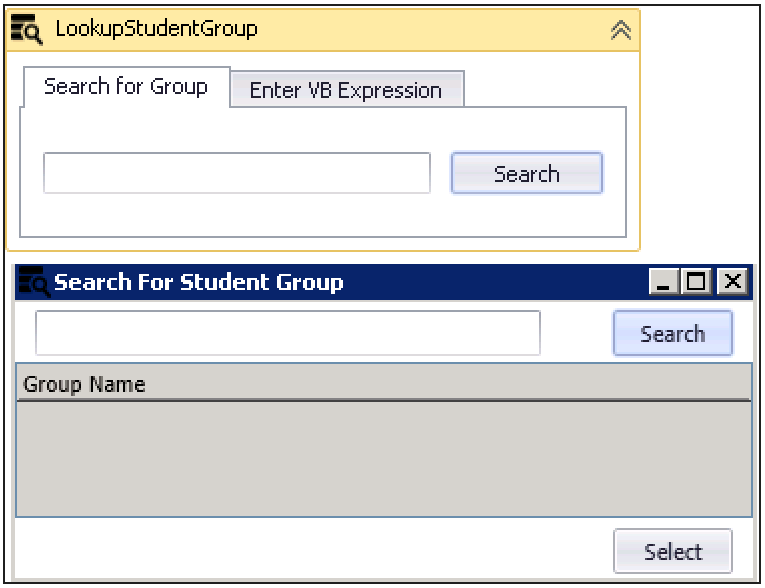
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Module	`InArgument<String>`	Yes	Select a value in the drop-down list of the activity in the Designer window.

Property	Value	Required	Notes
StudentAdvisors	`OutArgument <StudentAdvisorEntity[]>`	Yes	The LookupStudentAdvisors activity returns an array of student advisors associated with a Module and StudentEnrollmentPeriodId. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Common.Contracts > Cmc.Nexus.Common.Entities**, select **StudentAdvisorEntity**, and click **OK**.  Name Variable type FAAdvisor Cmc.Nexus.Common.Entities.StudentAdvisorEntity[] See StudentAdvisorEntity Class in the Anthology Student Object Library.
StudentEnrollmentPeriodId	`InArgument<Int32>`	Yes	Specify the Student Enrollment Period Id using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

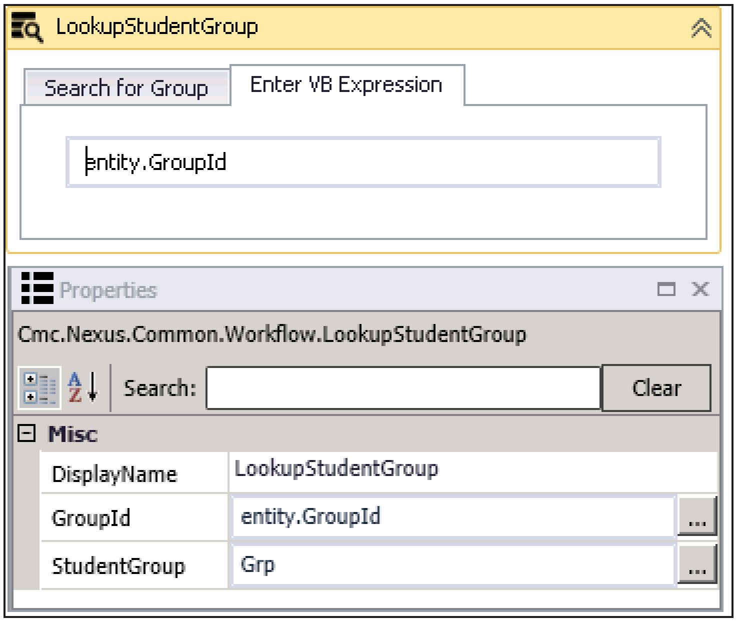
LookupStudentGroup (V2)

The LookupStudentGroup activity is a function that captures the Group Id from an event and returns the Group (name).

On the 'Search for Group' tab, click the Search button to find a student group.



Or, use the 'Enter VB Expression' tab to find a group.



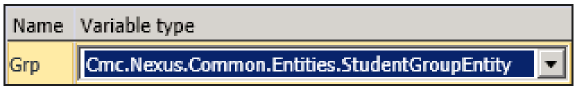
entity.GroupId

Cmc.Nexus.Common.Workflow.LookupStudentGroup

DisplayName	LookupStudentGroup
GroupId	entity.GroupId
StudentGroup	Grp

Properties

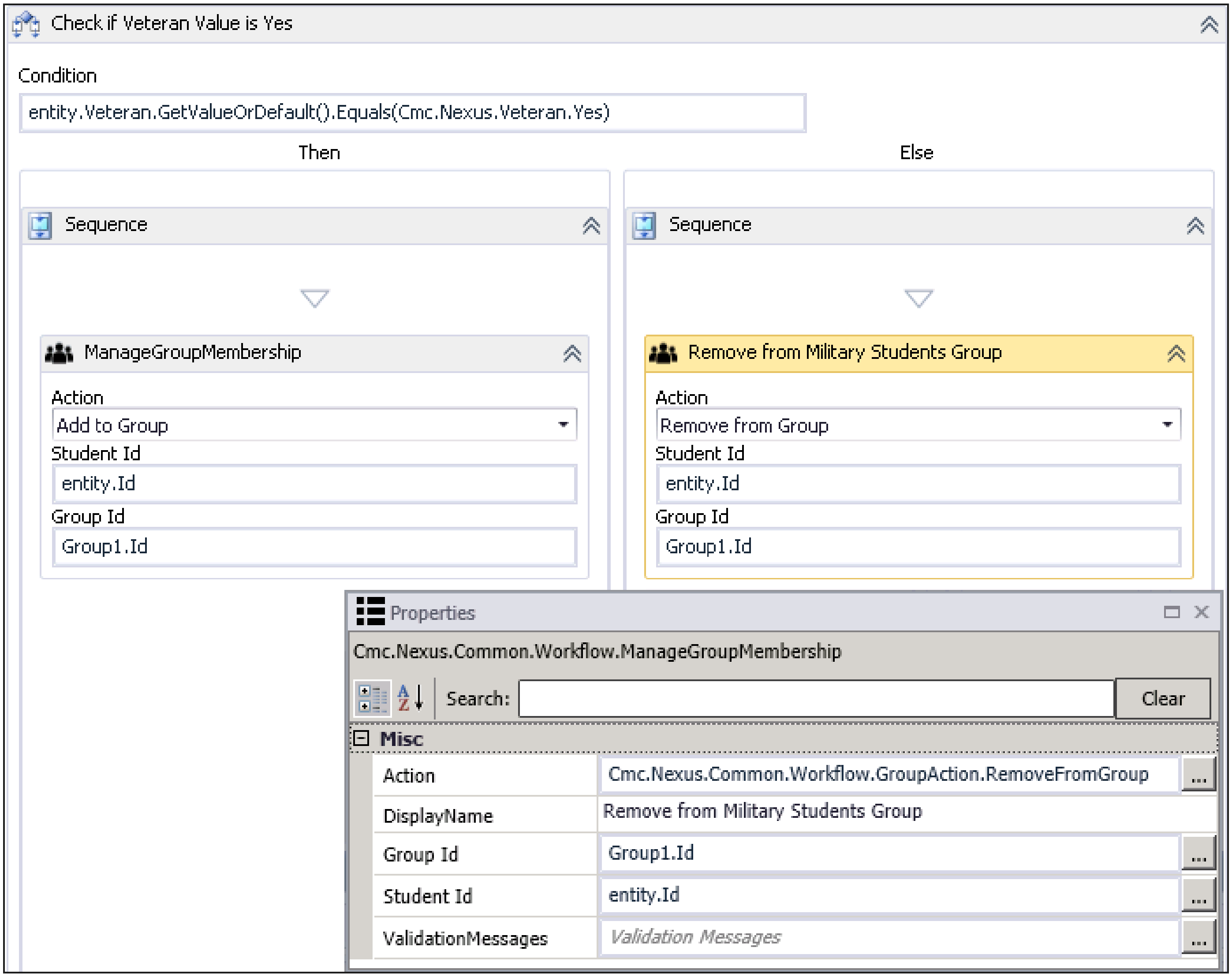
LookupStudentGroup Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
GroupId	`InArgument<Int32>`	Yes	Specify the Group Id captured from an event using a VB expression or variable.
StudentGroup	`OutArgument <StudentGroupEntity>`	Yes	The Student Group (name) returned by the lookup function, for example "Leads". This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Common.Contracts > Cmc.Nexus.Common.Entities**, select **StudentGroupEntity**, and click **OK**.  Name Variable type Grp Cmc.Nexus.Common.Entities.StudentGroupEntity See StudentGroupEntity Class in the Anthology Student Object Library.

ManageGroupMembership (V2)

The ManageGroupMembership activity enables you to automate the addition (or removal) of a student group member. The activity captures a Group Id and a Student Id from an event.

Use the LookupStudentGroup (V2) activity to capture the Group Id from an event and to identify the Group (name).



Check if Veteran Value is Yes

Condition: entity.Veteran.GetValueOrDefault().Equals(Cmc.Nexus.Veteran.Yes)

Then

Sequence

ManageGroupMembership

Action: Add to Group

Student Id: entity.Id

Group Id: Group1.Id

Else

Sequence

Remove from Military Students Group

Action: Remove from Group

Student Id: entity.Id

Group Id: Group1.Id

Properties

Cmc.Nexus.Common.Workflow.ManageGroupMembership

Action	Cmc.Nexus.Common.Workflow.GroupAction.RemoveFromGroup
DisplayName	Remove from Military Students Group
Group Id	Group1.Id
Student Id	entity.Id
ValidationMessages	*Validation Messages*

Properties

ManageGroupMembership Properties

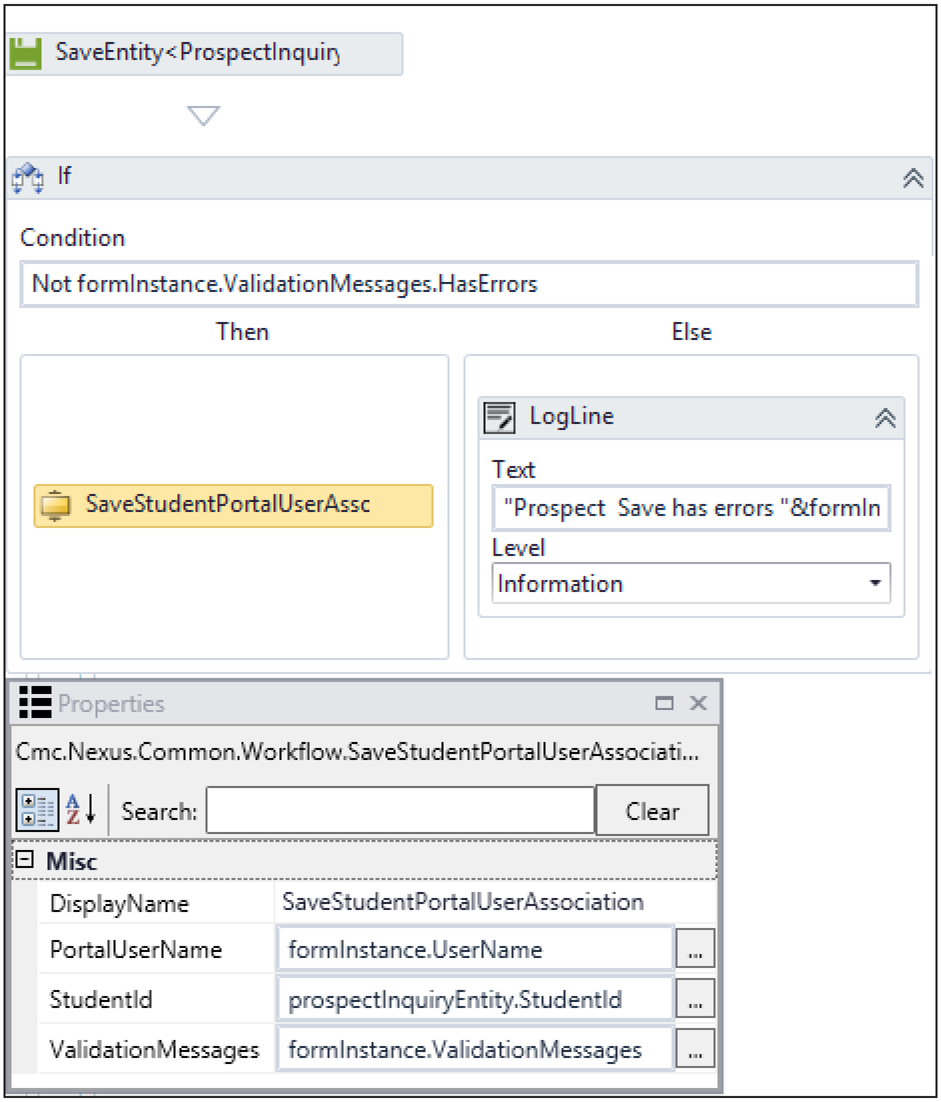
Property	Value	Required	Notes
Action	`InArgument<GroupAction>`	Yes	A drop-down list enabling you to select an action to take when the event occurs. The options are: • Add to Group See AddStudentToGroupRequest Class in the Anthology Student Object Library. • Remove from Group See RemoveStudentFromGroupRequest Class in the Anthology Student Object Library.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Group Id	`InArgument<Int32>`	Yes	The Group Id captured from an event.
Student Id	`InArgument<Int32>`	Yes	The Student Id captured from an event.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

SaveStudentPortalUserAssociation

The SaveStudentPortalUserAssociation activity creates a wpUserRelation record in the Portal database. The wpUserRelation record establishes a relation between a wpUser record in the Portal database and an syStudent record in the Anthology Student database. The wpUserRelation enables a student or staff user to log into the Portal. The record contains four columns for each user:

- wpUserID (Web Port User Id)
- Relation Type (Staff or Student)
- C2kID (Student ID)
- CampusID

The example below shows the SaveStudentPortalUserAssociation activity in a workflow for a form created in Forms Builder. The activity is triggered after a prospect enters his or her personal information. If the form is completed without errors and the PropectInquiry entity is saved, a wpUserRelation record is created and the prospect gains access to the Portal.



Condition: Not formInstance.ValidationMessages.HasErrors

Text: "Prospect Save has errors "&formIn

Level: Information

Cmc.Nexus.Common.Workflow.SaveStudentPortalUserAssociati...

DisplayName	SaveStudentPortalUserAssociation
PortalUserName	formInstance.UserName
StudentId	prospectInquiryEntity.StudentId
ValidationMessages	formInstance.ValidationMessages

Properties

SaveStudentPortalUserAssociation Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
PortalUserName	`InArgument<String>`	Yes	Specify the PortalUserName using a VB expression or variable.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

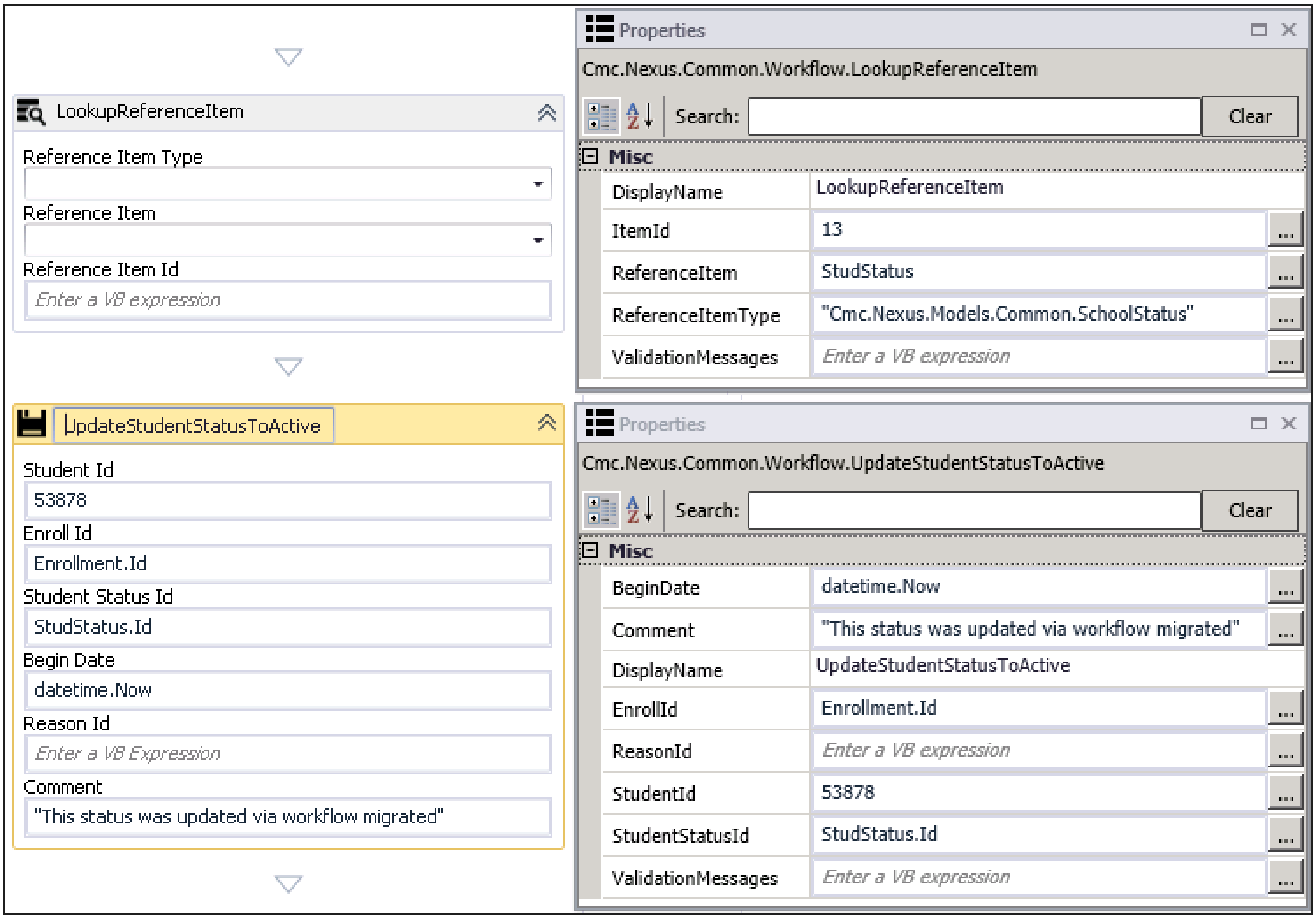
UpdateStudentStatusToActive (V2)

You can use the UpdateStudentStatusToActive activity to change the school status of a student to an Active (A) category so that you can trigger status changes when certain conditions occur.

For example, you could change the status 'Being Processed' or 'Temp Out' to 'Active' when specific events occur. You can use a LookupReferenceItem activity with "Reference Item Type = School Status" and "Reference Item = <status>" to find the status within the Active category that is to be changed in the workflow.

Status categories in Anthology Student are defined in the Setup > Status Codes > Status Codes tab. To determine Active category status values in the database, use the following SQL query:

```
Select S.Category, SS.*

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'A' Order by ss.Descrip
```



LookupReferenceItem

Reference Item Type

Reference Item

Reference Item Id

Enter a VB expression

UpdateStudentStatusToActive

Student Id: 53878

Enroll Id: Enrollment.Id

Student Status Id: StudStatus.Id

Begin Date: datetime.Now

Reason Id: *Enter a VB Expression*

Comment: "This status was updated via workflow migrated"

Properties

Cmc.Nexus.Common.Workflow.LookupReferenceItem

Search: Clear

Misc

DisplayName	LookupReferenceItem
ItemId	13
ReferenceItem	StudStatus
ReferenceItemType	"Cmc.Nexus.Models.Common.SchoolStatus"
ValidationMessages	*Enter a VB expression*

Properties

Cmc.Nexus.Common.Workflow.UpdateStudentStatusToActive

Search: Clear

Misc

BeginDate	datetime.Now
Comment	"This status was updated via workflow migrated"
DisplayName	UpdateStudentStatusToActive
EnrollId	Enrollment.Id
ReasonId	*Enter a VB expression*
StudentId	53878
StudentStatusId	StudStatus.Id
ValidationMessages	*Enter a VB expression*

Properties

UpdateStudentStatusToActive Properties

Property	Value	Required	Notes
BeginDate	`InArgument<DateTime>`	Yes	Specify a date using a VB expression or variable.
Comment	`InArgument<String>`	No	Specify a comment if applicable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EnrollId	`InArgument<Int32>`	Yes	Specify the Enroll Id using a VB expression or variable.
ReasonId	`InArgument<Int32>`	No	Specify the Reason Id using a VB expression or variable.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.

Property	Value	Required	Notes
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

UpdateStudentStatusToApplicant (V2)

You can use the UpdateStudentStatusToApplicant activity to change the school status of a student from a Lead or Applicant category to an Applicant Processing (C) category so that you can trigger status changes when certain conditions occur.

For example, you could change the status 'Being Processed' to 'Applicant' when a student is added to the Applicants groups. You can use a LookupReferenceItem activity with "Reference Item Type = School Status" and "Reference Item = <status>" to find the status within the Future Start category that is to be changed in the workflow.

Status categories in Anthology Student are defined in the Setup > Status Codes > Status Codes tab. To determine Applicant category status values in the database, use the following SQL query:

Select S.Category, SS.*

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'C' Order by ss.Descrip

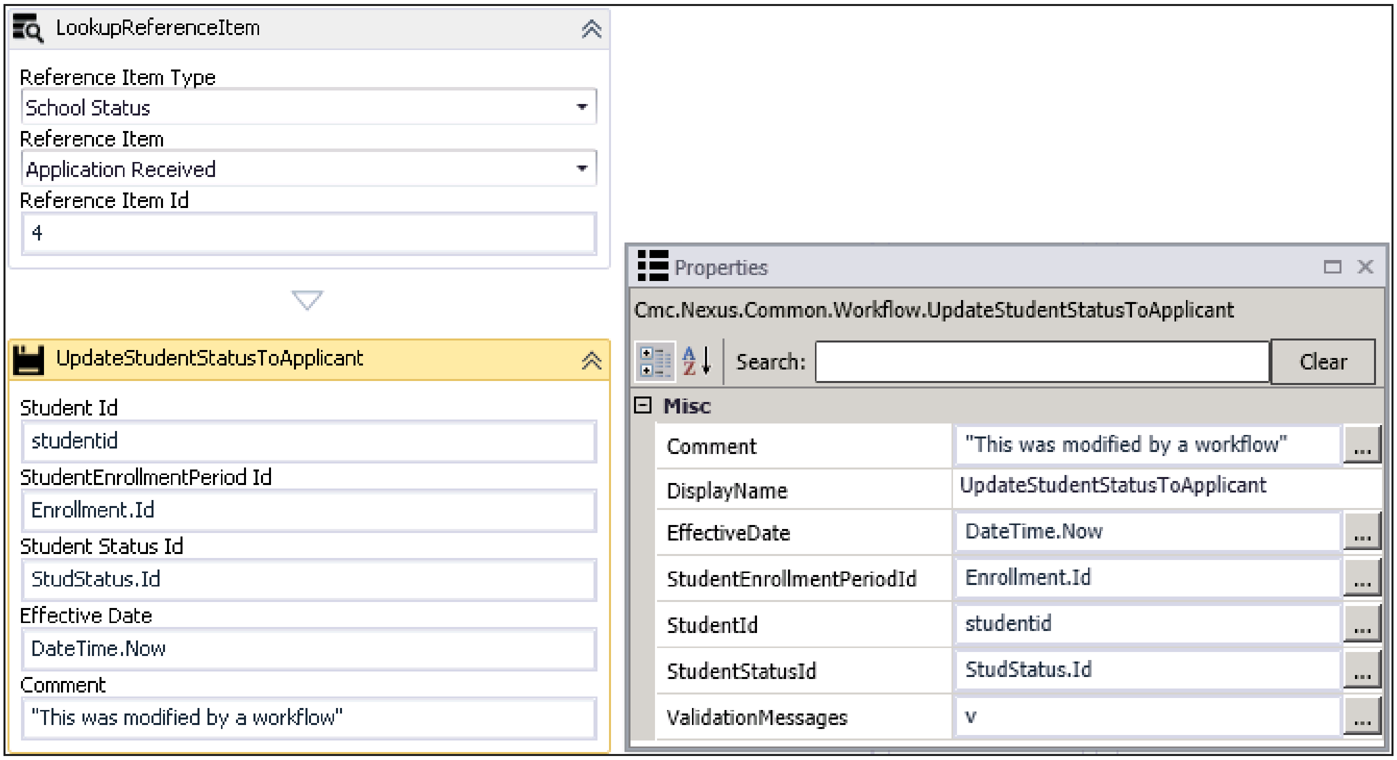
To determine StudentId values, use the following SQL query:

Select SyStudentId, syschoolstatusid, addenrollid, * from AdEnroll where SySchoolStatusID IN (Select SS.syschoolstatusid

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'C')



Reference Item Type
School Status

Reference Item
Application Received

Reference Item Id
4

UpdateStudentStatusToApplicant

Student Id
studentid

StudentEnrollmentPeriod Id
Enrollment.Id

Student Status Id
StudStatus.Id

Effective Date
DateTime.Now

Comment
"This was modified by a workflow"

Properties
Cmc.Nexus.Common.Workflow.UpdateStudentStatusToApplicant
Search:
Clear
Misc

Comment	"This was modified by a workflow"
DisplayName	UpdateStudentStatusToApplicant
EffectiveDate	DateTime.Now
StudentEnrollmentPeriodId	Enrollment.Id
StudentId	studentid
StudentStatusId	StudStatus.Id
ValidationMessages	v

Properties

UpdateStudentStatusToApplicant Properties

Property	Value	Required	Notes
Comment	`InArgument<String>`	No	Specify a comment if applicable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EffectiveDate	`InArgument<DateTime>`	Yes	Specify a date using a VB expression or variable. For example, to change the School Status whenever the event occurs, specify: DateTime.Now
StudentEnrollmentPeriodId	`InArgument<Int32>`	Yes	Specify the Student Enrollment Period Id using a VB expression or variable.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.

Property	Value	Required	Notes
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

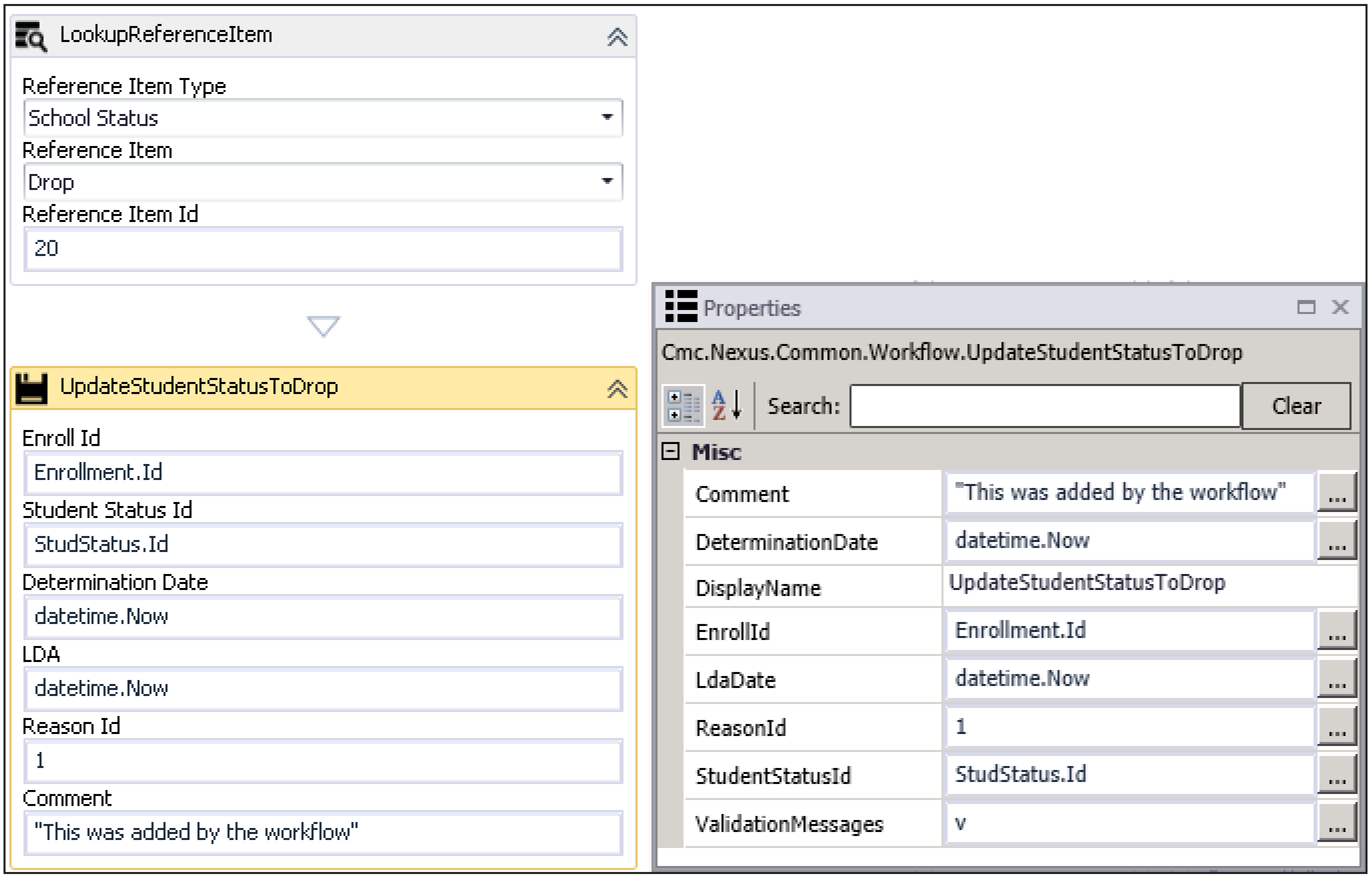
UpdateStudentStatusToDrop (V2)

You can use the UpdateStudentStatusToDrop activity to change the school status of a student to a Permanent Out (P) category so that you can trigger status changes when certain conditions occur.

For example, you could change the status 'Active' to 'Drop' when a student withdraws from all classes. You can use a LookupReferenceItem activity with "Reference Item Type = School Status" and "Reference Item = <status>" to find the status that is to be changed in the workflow.

Status categories in Anthology Student are defined in the Setup > Status Codes > Status Codes tab. To determine Permanent Out category status values in the database, use the following SQL query:

```
Select S.Category, SS.*

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'P' Order by ss.Descrip
```



LookupReferenceItem

Reference Item Type: School Status

Reference Item: Drop

Reference Item Id: 20

UpdateStudentStatusToDrop

Enroll Id: Enrollment.Id

Student Status Id: StudStatus.Id

Determination Date: datetime.Now

LDA: datetime.Now

Reason Id: 1

Comment: "This was added by the workflow"

Cmc.Nexus.Common.Workflow.UpdateStudentStatusToDrop

Misc

Comment	"This was added by the workflow"
DeterminationDate	datetime.Now
DisplayName	UpdateStudentStatusToDrop
EnrollId	Enrollment.Id
LdaDate	datetime.Now
ReasonId	1
StudentStatusId	StudStatus.Id
ValidationMessages	v

Properties

UpdateStudentStatusToDrop Properties

Property	Value	Required	Notes
Comment	`InArgument<String>`	No	Specify a comment if applicable.
DeterminationDate	`InArgument<DateTime>`	Yes	Specify the Determination Date using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EnrollId	`InArgument<Int32>`	Yes	Specify the Enroll Id using a VB expression or variable.
LdaDate	`InArgument<DateTime>`	Yes	Specify the Last Date of Attendance (LDA) using a VB expression or variable.
ReasonId	`InArgument<Int32>`	Yes	Specify the Reason Id using a VB expression or variable.

Property	Value	Required	Notes
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

UpdateStudentStatusToEnrolled (V2)

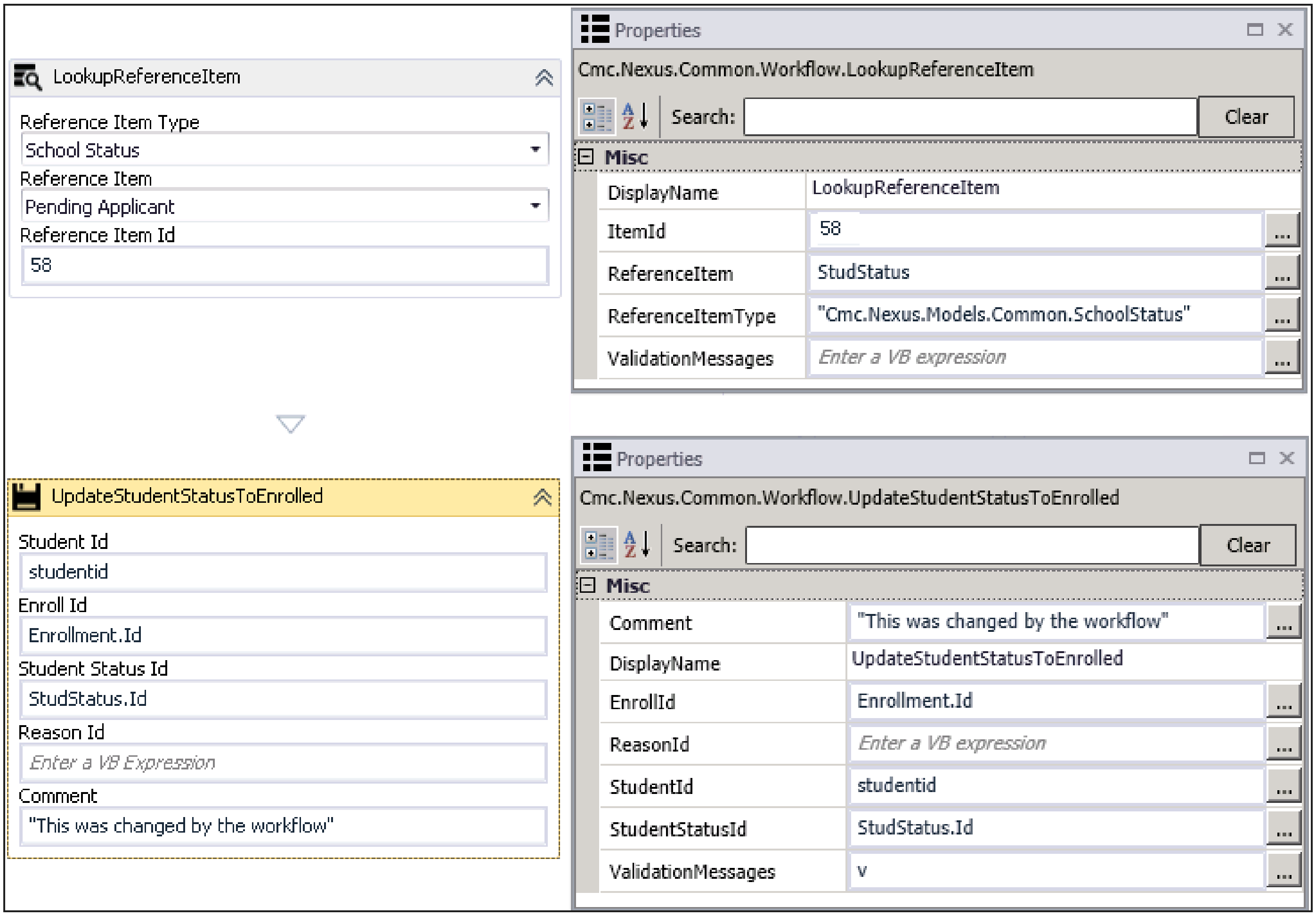
You can use the UpdateStudentStatusToEnrolled activity to change the school status of a student to an Enrolled (E) category so that you can trigger status changes when certain conditions occur.

For example, you could change the status 'Application Received' or 'Pending Applicant' to 'Enrolled' when specific events occur. You can use a LookupReferenceItem activity with "Reference Item Type = School Status" and "Reference Item = <status>" to find the status within the Enrolled category that is to be changed in the workflow.

Status categories in Anthology Student are defined in the Setup > Status Codes > Status Codes tab. To determine Enrolled category status values in the database, use the following SQL query:

```
Select S.Category, SS.*

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'E' Order by ss.Descrip
```

Note: You can update a student's status to NDS Enrolled Status (SyStatus.Category = 'X') using the Activities and Contracts package for Anthology Student 21.0.



LookupReferenceItem

Reference Item Type: School Status

Reference Item: Pending Applicant

Reference Item Id: 58

UpdateStudentStatusToEnrolled

Student Id: studentid

Enroll Id: Enrollment.Id

Student Status Id: StudStatus.Id

Reason Id: *Enter a VB Expression*

Comment: "This was changed by the workflow"

Properties

Cmc.Nexus.Common.Workflow.LookupReferenceItem

Misc	
DisplayName	LookupReferenceItem
ItemId	58
ReferenceItem	StudStatus
ReferenceItemType	"Cmc.Nexus.Models.Common.SchoolStatus"
ValidationMessages	*Enter a VB expression*

Properties

Cmc.Nexus.Common.Workflow.UpdateStudentStatusToEnrolled

Misc	
Comment	"This was changed by the workflow"
DisplayName	UpdateStudentStatusToEnrolled
EnrollId	Enrollment.Id
ReasonId	*Enter a VB expression*
StudentId	studentid
StudentStatusId	StudStatus.Id
ValidationMessages	v

Properties

UpdateStudentStatusToEnrolled Properties

Property	Value	Required	Notes
Comment	`InArgument<String>`	No	Specify a comment if applicable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EnrollId	`InArgument<Int32>`	Yes	Specify the Enroll Id using a VB expression or variable.
ReasonId	`InArgument<Int32>`	No	Specify the Reason Id using a VB expression or variable.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.

Property	Value	Required	Notes
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

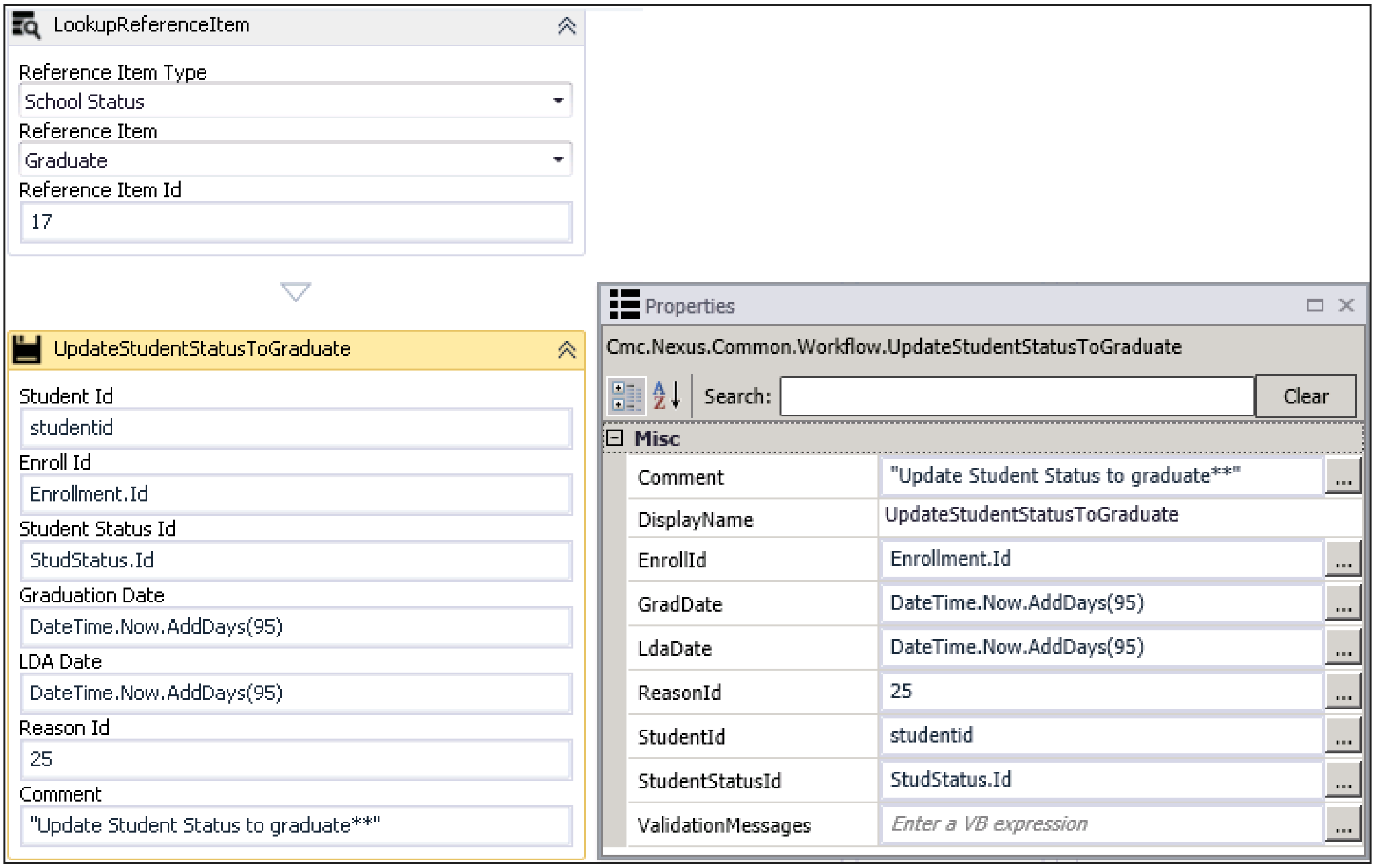
UpdateStudentStatusToGraduate (V2)

You can use the UpdateStudentStatusToGraduate activity to change the school status of a student to a Graduate (P - Permanent Out) category so that you can trigger status changes when certain conditions occur.

For example, you could change the status 'Active' to 'Graduate' when a student graduates. You can use a LookupReferenceItem activity with "Reference Item Type = School Status" and "Reference Item = <status>" to find the status within the Active category that is to be changed in the workflow.

Status categories in Anthology Student are defined in the Setup > Status Codes > Status Codes tab. To determine Graduate category status values in the database, use the following SQL query:

```
Select S.Category, SS.*

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'P' Order by ss.Descrip
```



LookupReferenceItem

Reference Item Type: School Status

Reference Item: Graduate

Reference Item Id: 17

UpdateStudentStatusToGraduate

Student Id: studentid

Enroll Id: Enrollment.Id

Student Status Id: StudStatus.Id

Graduation Date: DateTime.Now.AddDays(95)

LDA Date: DateTime.Now.AddDays(95)

Reason Id: 25

Comment: "Update Student Status to graduate**"

Properties

Cmc.Nexus.Common.Workflow.UpdateStudentStatusToGraduate

Search: Clear

Misc

Comment	"Update Student Status to graduate**"
DisplayName	UpdateStudentStatusToGraduate
EnrollId	Enrollment.Id
GradDate	DateTime.Now.AddDays(95)
LdaDate	DateTime.Now.AddDays(95)
ReasonId	25
StudentId	studentid
StudentStatusId	StudStatus.Id
ValidationMessages	Enter a VB expression

Properties

UpdateStudentStatusToGraduate Properties

Property	Value	Required	Notes
Comment	`InArgument<String>`	No	Specify a comment if applicable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EnrollId	`InArgument<Int32>`	Yes	Specify the Enroll Id using a VB expression or variable.
GradDate	`InArgument<DateTime>`	Yes	Specify the Graduation Date using a VB expression or variable.
LdaDate	`InArgument<DateTime>`	Yes	Specify the Last Date of Attendance (LDA) using a VB expression or variable.
ReasonId	`InArgument<Int32>`	No	Specify the Reason Id using a VB expression or variable.

Property	Value	Required	Notes
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

UpdateStudentStatusToLead (V2)

You can use the UpdateStudentStatusToLead activity to change the school status of a student in a Lead status to another Lead category (L - Lead) so that you can trigger status changes when certain conditions occur.

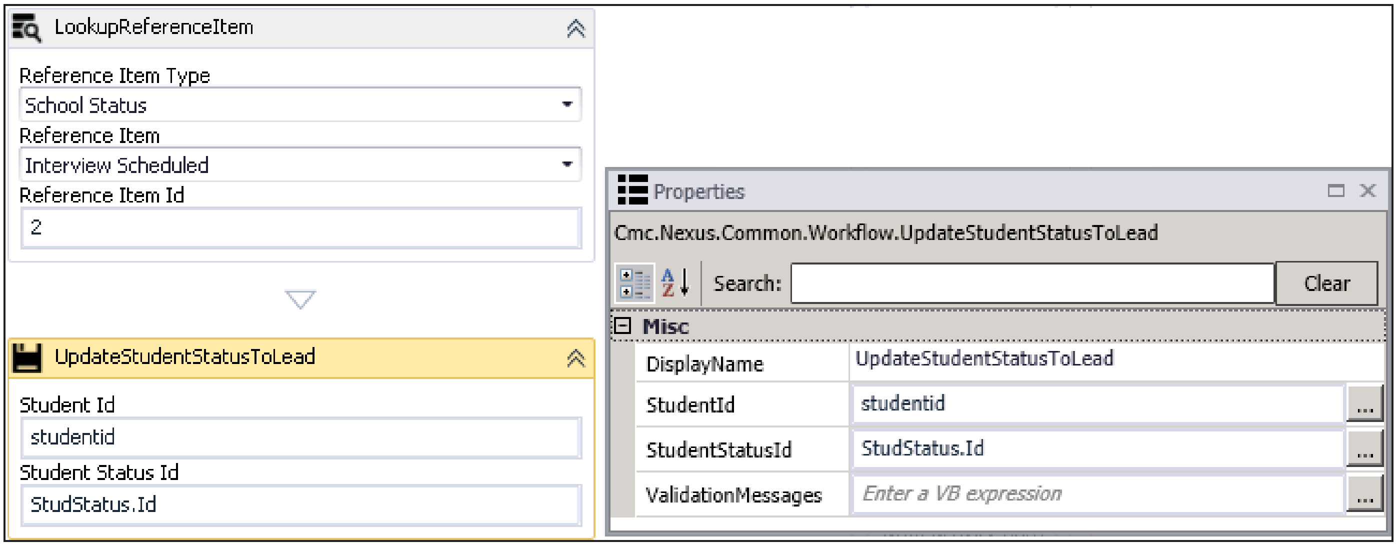
For example, you could change the status 'New Lead' to 'Interviewed' when a student is added to the Applicants groups. You can use a LookupReferenceItem activity with "Reference Item Type = School Status" and "Reference Item = <status>" to find the status within the Lead category that is to be changed in the workflow.

Status categories in Anthology Student are defined in the Setup > Status Codes > Status Codes tab. To determine Lead category status values in the database, use the following SQL query:

```
Select S.Category, SS.*

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'L' Order by ss.Descrip
```

To determine StudentId values, use the following SQL query:

```
Select SyStudentId,  syschoolstatusid, adenrollid, * from AdEnroll where SySchoolStatusID IN (Select SS.syschoolstatusid

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatuSID = S.SyStatusID

Where S.Category = 'L')
```


LookupReferenceItem

Reference Item Type: School Status

Reference Item: Interview Scheduled

Reference Item Id: 2

UpdateStudentStatusToLead

Student Id: studentid

Student Status Id: StudStatus.Id

Properties

Cmc.Nexus.Common.Workflow.UpdateStudentStatusToLead

Search:

Clear

Misc

DisplayName	UpdateStudentStatusToLead
StudentId	studentid
StudentStatusId	StudStatus.Id
ValidationMessages	*Enter a VB expression*

Properties

UpdateStudentStatusToLead Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

UpdateStudentStatusToTempOut (V2)

You can use the UpdateStudentStatusToTempOut activity to change the school status of a student to a Temporary Out (T) category so that you can trigger status changes when certain conditions occur.

For example, you could change the status 'Active' to 'Temporary Out' when a student requests a medical leave. You can use a LookupReferenceItem activity with "Reference Item Type = School Status" and "Reference Item = <status>" to find the status within the Active category that is to be changed in the workflow.

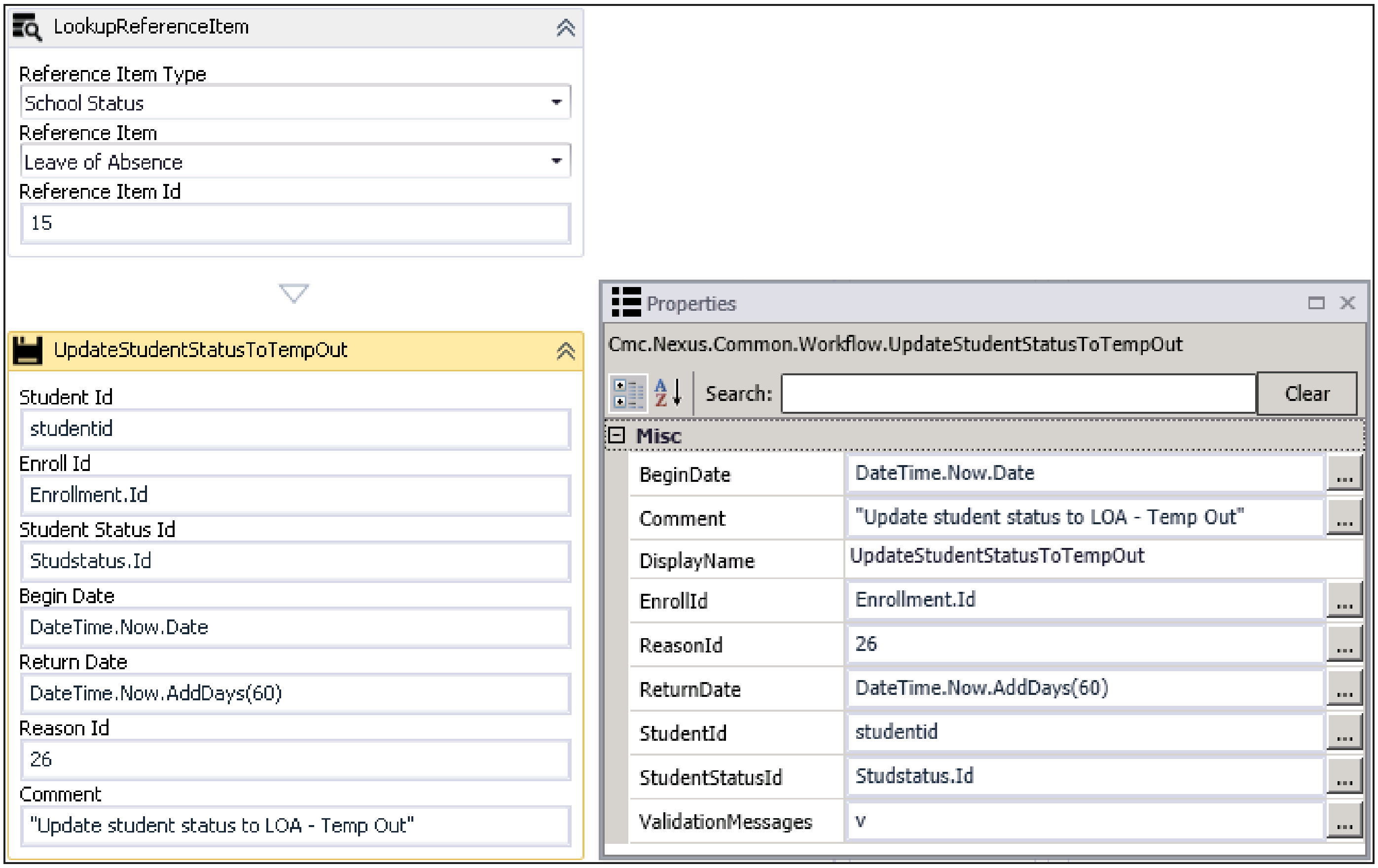
Status categories in Anthology Student are defined in the Setup > Status Codes > Status Codes tab. To determine Temporary Out category status values in the database, use the following SQL query:

```
Select S.Category, SS.*
```

from SySchoolStatus SS(nolock)

JOIN SyStatus S(nolock) ON SS.SyStatusID = S.SyStatusID

Where S.Category = 'T' Order by ss.Descrip



LookupReferenceItem

Reference Item Type: School Status

Reference Item: Leave of Absence

Reference Item Id: 15

UpdateStudentStatusToTempOut

Student Id: studentid

Enroll Id: Enrollment.Id

Student Status Id: Studstatus.Id

Begin Date: DateTime.Now.Date

Return Date: DateTime.Now.AddDays(60)

Reason Id: 26

Comment: "Update student status to LOA - Temp Out"

Properties

Cmc.Nexus.Common.Workflow.UpdateStudentStatusToTempOut

Search: Clear

Misc

BeginDate	DateTime.Now.Date
Comment	"Update student status to LOA - Temp Out"
DisplayName	UpdateStudentStatusToTempOut
EnrollId	Enrollment.Id
ReasonId	26
ReturnDate	DateTime.Now.AddDays(60)
StudentId	studentid
StudentStatusId	Studstatus.Id
ValidationMessages	v

Properties

UpdateStudentStatusToTempOut Properties

Property	Value	Required	Notes
BeginDate	`InArgument<DateTime>`	Yes	Specify the Begin Date of the Temporary Out status using a VB expression or variable.
Comment	`InArgument<String>`	No	Specify a comment if applicable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EnrollId	`InArgument<Int32>`	Yes	Specify the Enroll Id using a VB expression or variable.

Property	Value	Required	Notes
ReasonId	`InArgument<Int32>`	No	Specify the Reason Id using a VB expression or variable.
ReturnDate	`InArgument<DateTime>`	Yes	Specify the Return Date using a VB expression or variable.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
StudentStatusId	`InArgument<Int32>`	Yes	Specify the Student Status Id using a VB expression or variable.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

Cmc.Nexus.Crm.Workflow

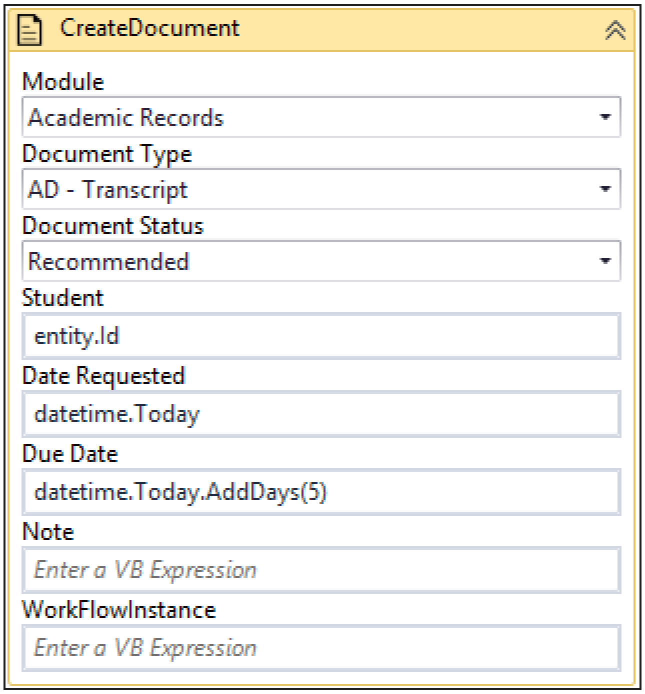
CreateDocument (V2)

The CreateDocument activity enables you to create a student document in a workflow. The CreateDocument activity is typically used in conjunction with a LookupReferenceItem activity to retrieve the Document Type associated with a Document Type Id.

This activity creates an instance of a Document; it does not save it to the database. To persist the Document in the database, insert a SaveDocument (V2) activity.

Note: The Activities and Contracts packages for Anthology Student version 18.0.2 and later modify the CreateDocument (V2) activity as follows:

- The Module selection is no longer required. The Module Id is derived from the selected Document Type. The Module field is retained in the user interface for backward compatibility only.
- It is no longer necessary to use Assign activities for the DocumentImage, OriginalFileName, ImageType, and IsDocumentAddedManually properties.



CreateDocument

Module: Academic Records

Document Type: AD - Transcript

Document Status: Recommended

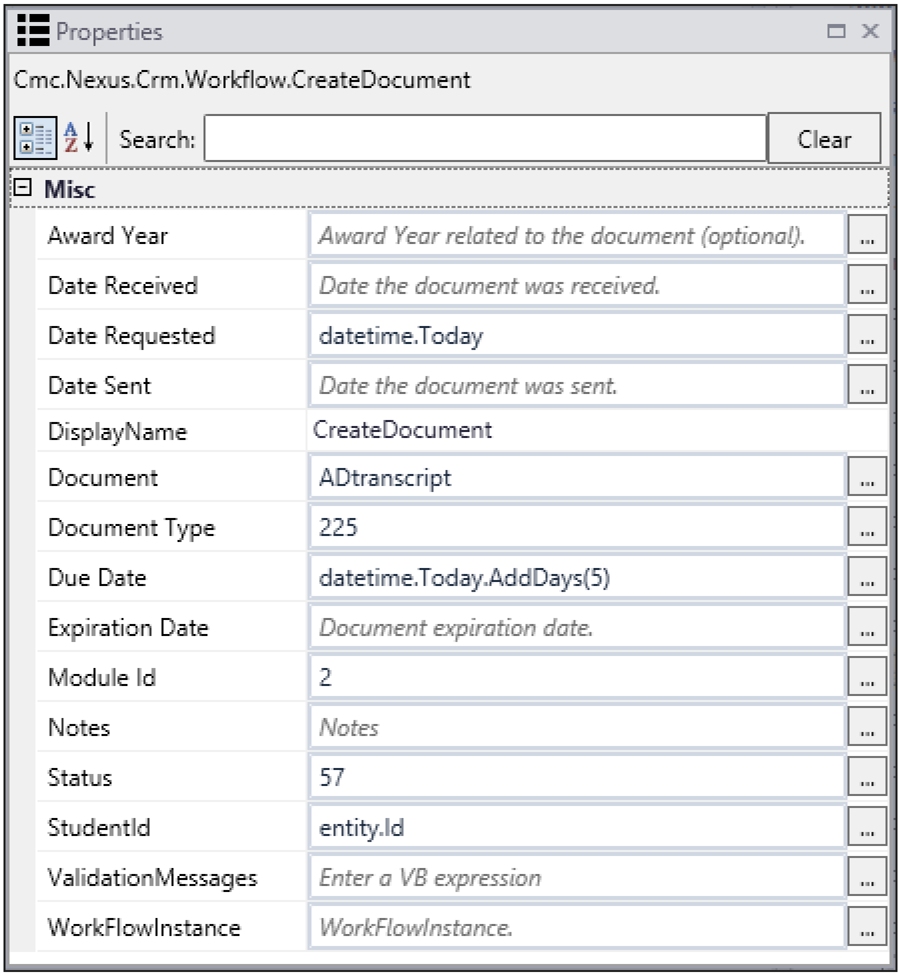
Student: entity.Id

Date Requested: datetime.Today

Due Date: datetime.Today.AddDays(5)

Note: *Enter a VB Expression*

WorkFlowInstance: *Enter a VB Expression*



Properties

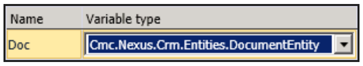
Cmc.Nexus.Crm.Workflow.CreateDocument

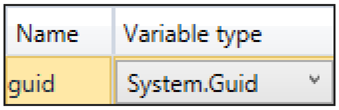
Award Year	*Award Year related to the document (optional).*
Date Received	*Date the document was received.*
Date Requested	datetime.Today
Date Sent	*Date the document was sent.*
DisplayName	CreateDocument
Document	ADtranscript
Document Type	225
Due Date	datetime.Today.AddDays(5)
Expiration Date	*Document expiration date.*
Module Id	2
Notes	*Notes*
Status	57
StudentId	entity.Id
ValidationMessages	*Enter a VB expression*
WorkFlowInstance	*WorkFlowInstance.*

Properties

CreateDocument Properties

Property	Value	Required	Notes
Award Year	`InArgument<String>`	No	Award Year related to the document.
Date Received	`InArgument<Nullable<DateTime>>`	No	Specify a date using a VB expression or variable. For example, to create the document whenever the event occurs, specify: `DateTime.Now`
Date Requested	`InArgument<Nullable<DateTime>>`	Yes	Specify a date using a VB expression or variable.
Date Sent	`InArgument<Nullable<DateTime>>`	No	Specify a date using a VB expression or variable.

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Document	`OutArgument<DocumentEntity>`	Yes	This is a variable that can be used in subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Crm.Contracts . Cmc.Nexus.Crm.Entities**, select **DocumentEntity** and click **OK**.  Name Variable type Doc Cmc.Nexus.Crm.Entities.DocumentEntity See DocumentEntity Class in the Anthology Student Object Library.
Document Type	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.
Due Date	`InArgument<Nullable<DateTime>>`	No	Specify a date using a VB expression or variable. For example, to specify a due date that is 30 days after the event occurred, specify: `DateTime.Now.AddDays (30)`
Expiration Date	`InArgument<Nullable<DateTime>>`	No	Specify a date using a VB expression or variable.
Module Id	`InArgument<Int32>`	No	The Module Id is derived automatically from the Document Type selection.
Notes	`InArgument<String>`	No	Specify a note related to the Document being created.

Property	Value	Required	Notes
Status	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.
StudentId	`InArgument<Int32>`	No	Specify a Student Id using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.
WorkFlowInstance	`InArgument<Guid>`	No	Specify the Id associated with the workflow instance to resume using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **mscorlib > System**, select **Guid**, and click **OK**.  Name Variable type guid System.Guid To remove a WorkflowInstance value, see Clear a Workflow Instance Id.

CreateTask (V2)

⚠ **Prerequisite**: When this activity is used with Anthology Student 21.2.0 and later, the APIUser must have authorization to access to the entity requested in the OData query. For more information, see Security Enhancement for OData Queries.

The CreateTask activity enables you to create an Anthology Student Contact Manager activity, a CampusNexus CRM Interaction, an appointment, or a notification.

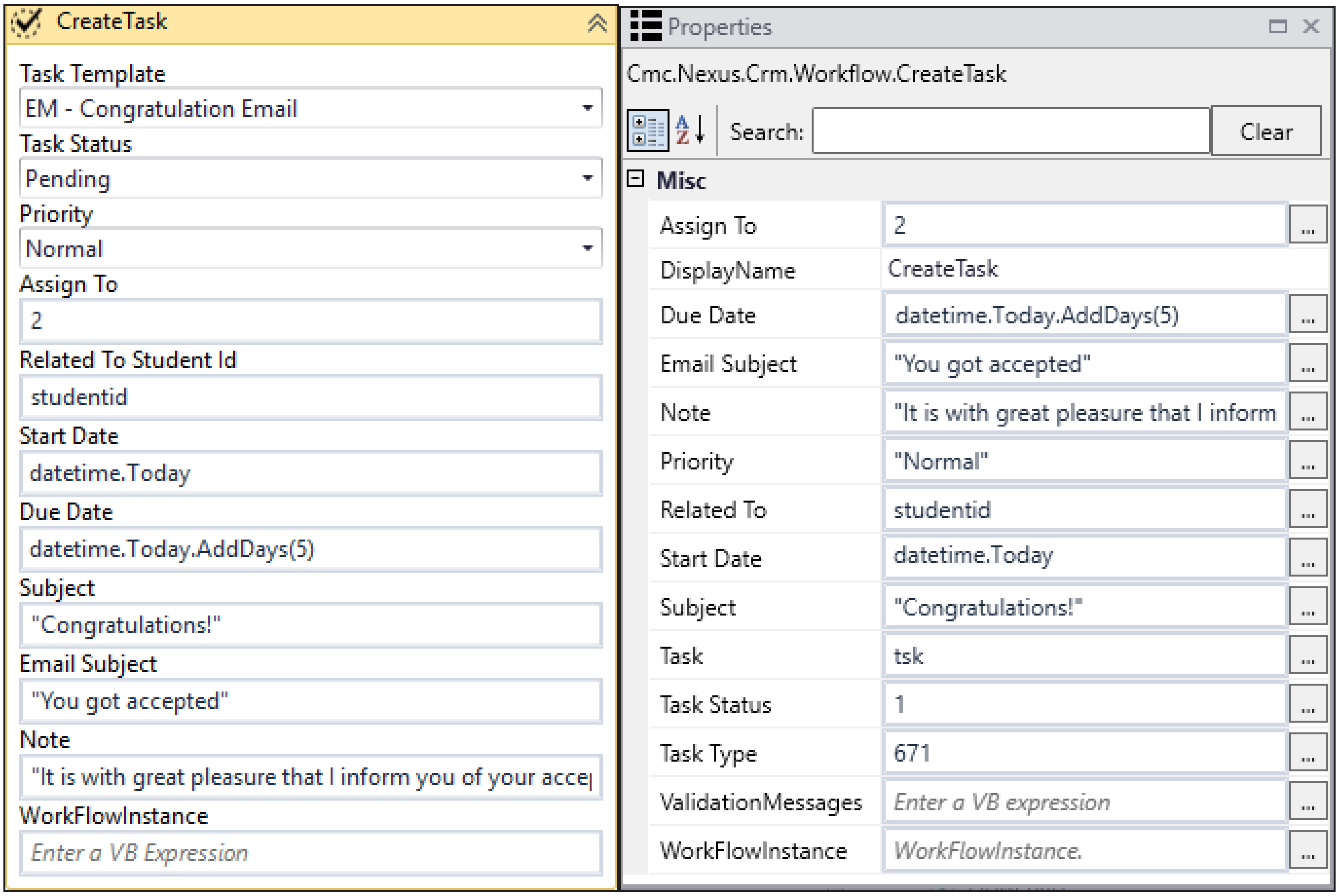
The out argument 'Task' is a variable that calls the newTask() function. The newTask() function can be used in workflows for multiple applications, such as Anthology Student and Anthology CRM.

The CreateTask activity creates an instance of a Task; it does not save the Task to the database. The workflow can include other activities that manipulate the Task before it is saved. To persist the Task in the database, insert a SaveTask (V2) activity.

Notes:

- In Workflow Composer 3.0 with Anthology Student 21.0 and later, the "Email Subject" property is added to the CreateTask activity.
- Anthology Student 21.2 and later uses the Kendo library instead of Moment.js to format the **`<DateTime>`** property. The differences between Kendo and Moment.js affect the code used to return current date and time values.
  - `DateTime.`**`Now`** (Moment.js) returns the current date and time, e.g., 2011-07-01 10:09.45310.
  - `DateTime.`**`Today`** (Kendo) returns the current date with the time components set to zero, e.g., 2011-07-01 00:00.00000.

  If you receive validation errors related to date and time values in your workflows, replace `DateTime.Now` with `DateTime.Today`.



CreateTask

Task Template: EM - Congratulation Email

Task Status: Pending

Priority: Normal

Assign To: 2

Related To Student Id: studentid

Start Date: datetime.Today

Due Date: datetime.Today.AddDays(5)

Subject: "Congratulations!"

Email Subject: "You got accepted"

Note: "It is with great pleasure that I inform you of your acce|

WorkFlowInstance: *Enter a VB Expression*

Properties

Cmc.Nexus.Crm.Workflow.CreateTask

Search: Clear

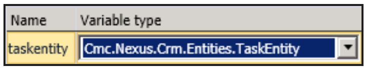
Misc

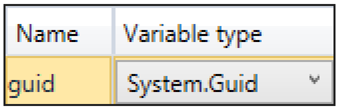
Assign To	2
DisplayName	CreateTask
Due Date	datetime.Today.AddDays(5)
Email Subject	"You got accepted"
Note	"It is with great pleasure that I inform
Priority	"Normal"
Related To	studentid
Start Date	datetime.Today
Subject	"Congratulations!"
Task	tsk
Task Status	1
Task Type	671
ValidationMessages	*Enter a VB expression*
WorkFlowInstance	WorkFlowInstance.

Properties

CreateTask Properties

Property	Value	Required	Notes
Assign To	`InArgument<Int32>`	Yes	Specify the Owner User Id using a VB expression or variable.
Email Subject	`InArgument<string>`	No	Enter a string that indicates the email subject.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Due Date	`InArgument<DateTime>`	Yes	Specify a date using a VB expression or variable.

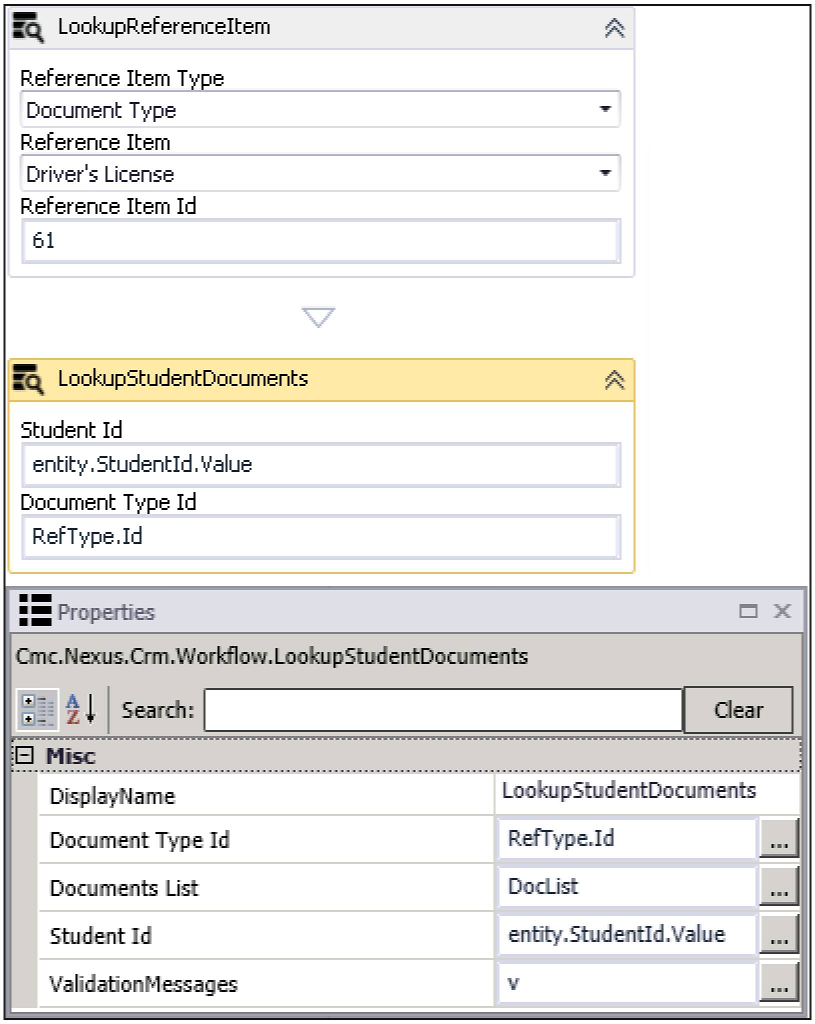
Property	Value	Required	Notes
Note	`InArgument<String>`	No	Specify a note related to the Task using a VB expression or variable, for example: `"Check out" & entity.FirstName & " "& entity.LastName`
Priority	`InArgument<TaskPriority>`	Yes	Select a value in the drop-down list of the activity in the Designer window.
Related To	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
Start Date	`InArgument<DateTime>`	No	The time the Task is scheduled to begin. Only the time portion of this value is relevant. Specify a value using a VB expression or variable.
Subject	`InArgument<String>`	Yes	Enter a string that indicates the Task subject.
Task	`OutArgument<TaskEntity>`	Yes	This is a variable that can be used in subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Crm.Contracts . Cmc.Nexus.Crm.Entities**, select **TaskEntity**, and click **OK**.  Name Variable type taskentity Cmc.Nexus.Crm.Entities.TaskEntity See TaskEntity Class in the Anthology Student Object Library.
Task Status	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.

Property	Value	Required	Notes
Task Template	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window. The drop-down list retrieves values from the CmTemplate table. If you know the Task Template Id, specify the Id value in the Properties pane.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.
WorkFlowInstance	`InArgument<Guid>`	No	Specify the Id associated with the workflow instance to resume using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **mscorlib > System**, select **Guid**, and click **OK**.  Name Variable type guid System.Guid To remove a WorkflowInstance value, see Clear a Workflow Instance Id.

LookupStudentDocuments

The LookupStudentDocuments activity returns the documents associated with a particular student. You can use this activity to modify the attributes of a Student Document using a workflow.

For example, you can look up a Document Type using a LookupReferenceItem activity and then use a LookupStudentDocuments activity to look up the students to whom the document has been associated via a Contact Manager activity. Based on an event, you can then change the document status or perform other activities, e.g., close a Contact Manager activity.



LookupReferenceItem

Reference Item Type: Document Type

Reference Item: Driver's License

Reference Item Id: 61

LookupStudentDocuments

Student Id: entity.StudentId.Value

Document Type Id: RefType.Id

Properties

Cmc.Nexus.Crm.Workflow.LookupStudentDocuments

Search:

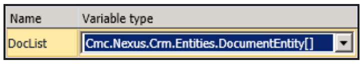
Misc

DisplayName	LookupStudentDocuments
Document Type Id	RefType.Id
Documents List	DocList
Student Id	entity.StudentId.Value
ValidationMessages	v

Properties

LookupStudentDocuments Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

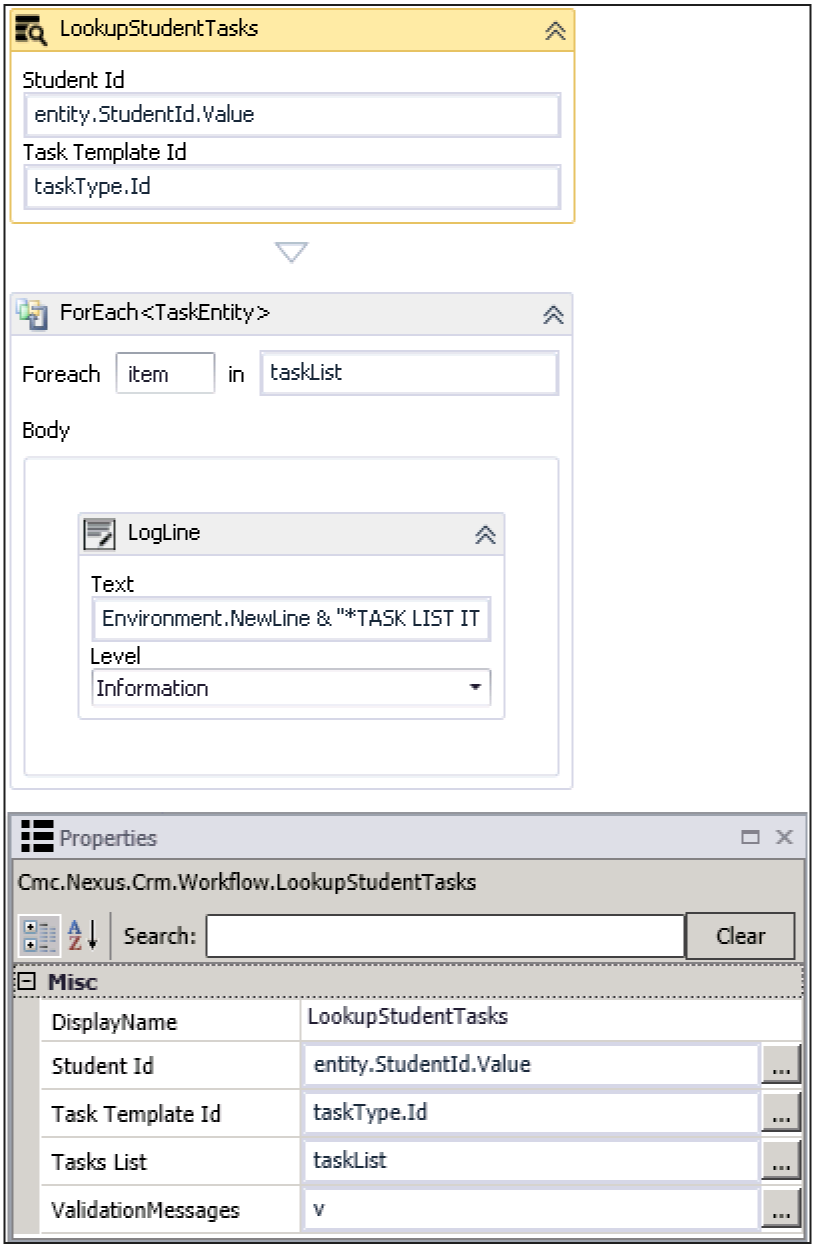
Property	Value	Required	Notes
Document Type Id	`InArgument<Int32>`	Yes	The DocumentTypeId captured from an event.
Documents List	`OutArgument<DocumentEntity[]>`	Yes	The Document returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Crm.Contract > Cmc.Nexus.Crm.Entities**, select **DocumentEntity**, and click **OK**.  Name Variable type DocList Cmc.Nexus.Crm.Entities.DocumentEntity[] See DocumentEntity Class in the Anthology Student Object Library.
Student Id	`InArgument<Int32>`	Yes	The Student Id captured from an event.

ValidationMessages	`OutArgument`
	`<ValidationMessageCollection>`
		No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

LookupStudentTasks (V2)

The LookupStudentTasks activity returns the Student Tasks associated with a particular student. You can use this activity to modify the attributes of a Student Task using a workflow.

For example, you can look up a task (Contact Manager activity) that has already been associated with a student and based on an event and change the status or result of an activity using a workflow.



Student Id: entity.StudentId.Value

Task Template Id: taskType.Id

Foreach item in taskList

Text: Environment.NewLine & "*TASK LIST IT

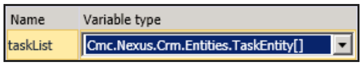
Level: Information

Cmc.Nexus.Crm.Workflow.LookupStudentTasks

DisplayName	LookupStudentTasks
Student Id	entity.StudentId.Value
Task Template Id	taskType.Id
Tasks List	taskList
ValidationMessages	v

Properties

LookupStudentTasks Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Student Id	`InArgument<Int32>`	Yes	The Student ID captured from an event.
Task Template Id	`InArgument<Int32>`	No	The TaskTemplateId captured from an event. If this property is left blank, all tasks are returned.
Task List	`OutArgument<TaskEntity[]>`	Yes	The Task returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Crm.Contract > Cmc.Nexus.Crm.Entities**, select **TaskEntity**, and click **OK**.  Name Variable type taskList Cmc.Nexus.Crm.Entities.TaskEntity[] See TaskEntity Class in the Anthology Student Object Library.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

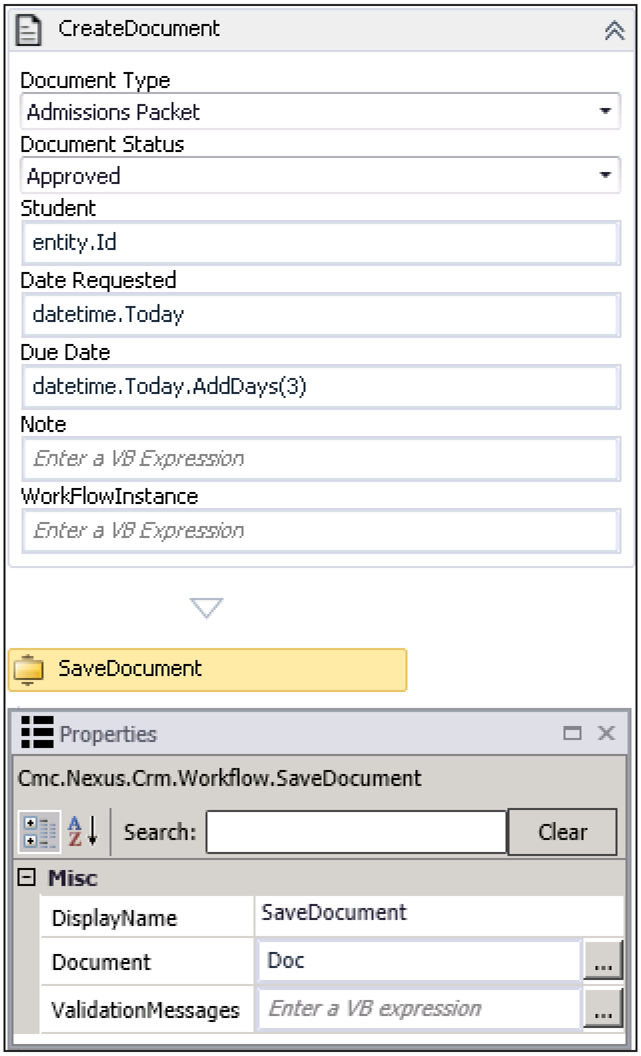
SaveDocument (V2)

The SaveDocument activity enables you to save a document record (INSERT mode). The document will be associated with a Person record.

You can also use this activity to modify an existing document record (UPDATE mode).

Notes:

- You can modify the following fields using the SaveDocument activity:
    - ApprovalDate
    - DocumentStatusId
    - DueDate
    - ExperiationDate
    - Note
    - ReceivedDate
    - RequestDate
    - SentDate
- If you update the PersonId, the StudentId or ProspectId must be updated as well because these fields reference the same student (SyStudentId).
- You cannot delete existing values (that is, fields that have a value cannot be set to NULL).



CreateDocument

Document Type: Admissions Packet

Document Status: Approved

Student: entity.Id

Date Requested: datetime.Today

Due Date: datetime.Today.AddDays(3)

Note: *Enter a VB Expression*

WorkFlowInstance: *Enter a VB Expression*

SaveDocument

Properties

Cmc.Nexus.Crm.Workflow.SaveDocument

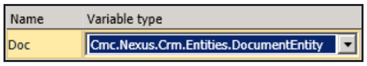
Misc

DisplayName	SaveDocument
Document	Doc
ValidationMessages	*Enter a VB expression*

Properties

SaveDocument Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
Document	`InOutArgument<DocumentEntity>`	Yes	Specify the Document using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Crm.Contracts . Cmc.Nexus.Crm.Entities**, select **DocumentEntity**, and click **OK**.  Name Variable type Doc Cmc.Nexus.Crm.Entities.DocumentEntity See DocumentEntity Class in the Anthology Student Object Library.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

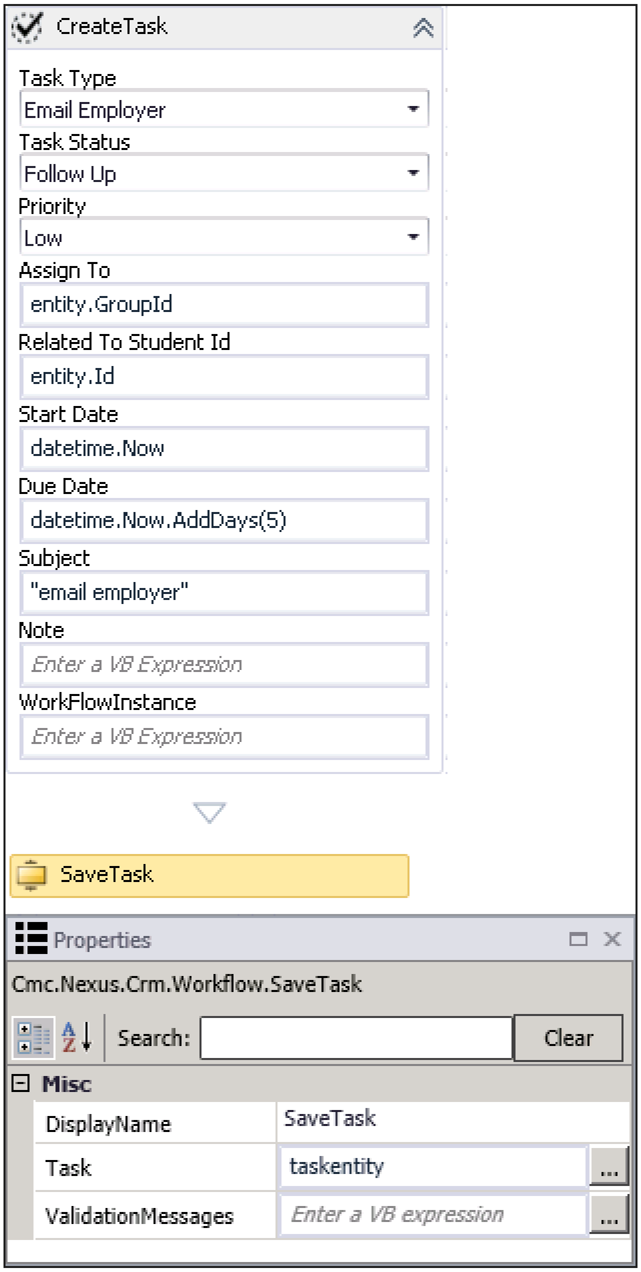
SaveTask (V2)

The SaveTask activity enables you to save a Task (INSERT mode) and display a validation message.

SaveTask is used after a CreateTask (V2) activity has created a Task instance. Save Task will persist a Task instance in the database by calling the API.

You can also use this activity to modify an existing task record (UPDATE mode). The following fields can be updated (corresponding Contact Manager Service API fields in parenthesis):

- DueDate
- Note (Comments)
- OwnerUserId (AssignedStaffId)
- Priority
- StartDate
- Subject
- TaskResultId (ActivityResultId)
- TaskStatusId (ActivityStatusId)



CreateTask

Task Type: Email Employer

Task Status: Follow Up

Priority: Low

Assign To: entity.GroupId

Related To Student Id: entity.Id

Start Date: datetime.Now

Due Date: datetime.Now.AddDays(5)

Subject: "email employer"

Note: Enter a VB Expression

WorkFlowInstance: Enter a VB Expression

SaveTask

Properties

Cmc.Nexus.Crm.Workflow.SaveTask

Search:

Clear

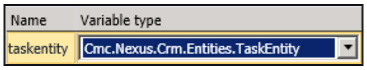
Misc

DisplayName	SaveTask
Task	taskentity
ValidationMessages	Enter a VB expression

Properties

SaveTask Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

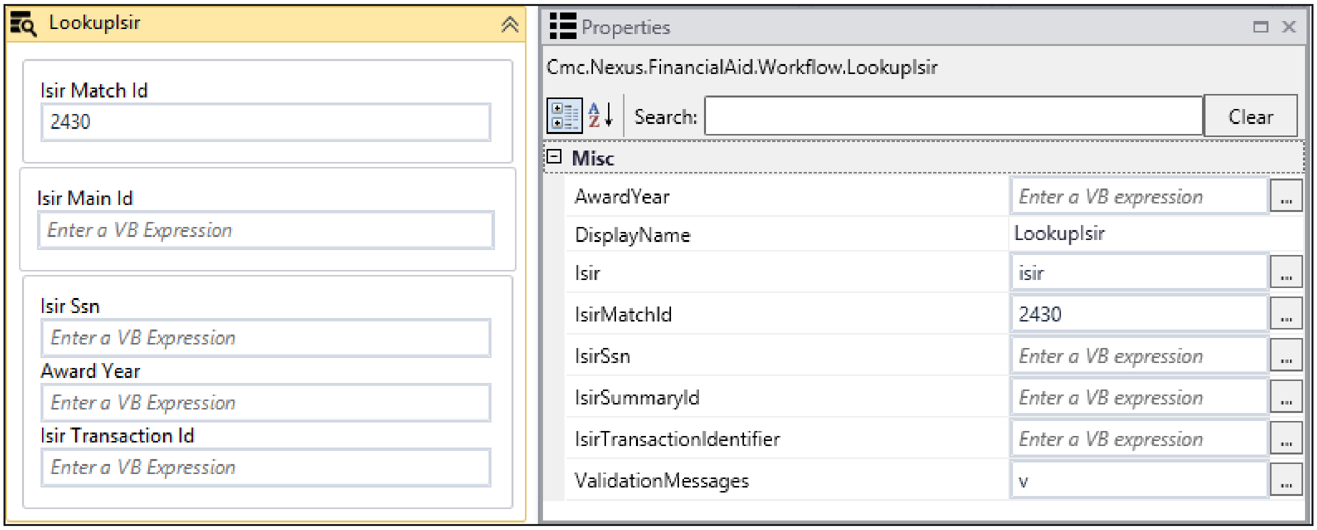
Property	Value	Required	Notes
Task	`InOutArgument<TaskEntity>`	Yes	Specify the entity to be saved using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types....** In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.Crm.Contracts . Cmc.Nexus.Crm.Entities**, select **TaskEntity**, and click **OK**.  Name Variable type taskentity Cmc.Nexus.Crm.Entities.TaskEntity See TaskEntity Class in the Anthology Student Object Library.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

LookupIsir

The LookupIsir activity returns all fields in the Institutional Student Information Records (ISIR) entity. This activity enables you to create workflows around ISIR specific events.

The optional input values of the LookupIsir activity can be used as follows:

- LookupIsir based on **ISIR Match Id** (Conditionally required input value marked C1 in the table below)



LookupIsir

Isir Match Id: 2430

Isir Main Id: Enter a VB Expression

Isir Ssn: Enter a VB Expression

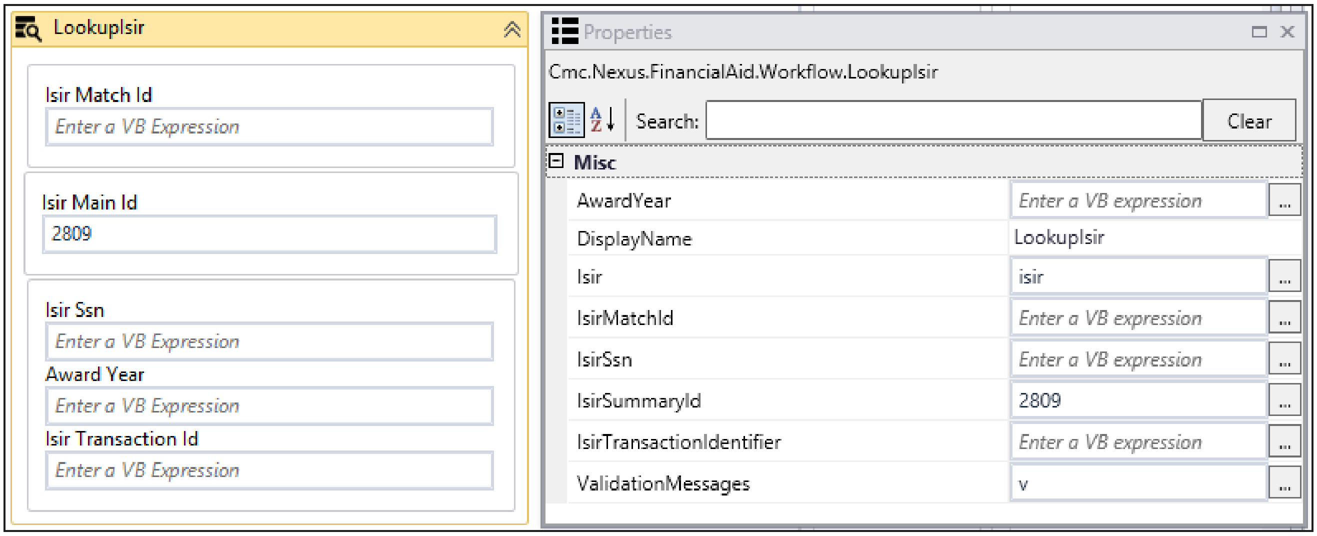
Award Year: Enter a VB Expression

Isir Transaction Id: Enter a VB Expression

Cmc.Nexus.FinancialAid.Workflow.LookupIsir

AwardYear	Enter a VB expression
DisplayName	LookupIsir
Isir	isir
IsirMatchId	2430
IsirSsn	Enter a VB expression
IsirSummaryId	Enter a VB expression
IsirTransactionIdentifier	Enter a VB expression
ValidationMessages	v

- LookupIsir based on **ISIR Main Id** (IsirSummaryId property) (Conditionally required input value marked C2 in the table below)



LookupIsir

Isir Match Id: Enter a VB Expression

Isir Main Id: 2809

Isir Ssn: Enter a VB Expression

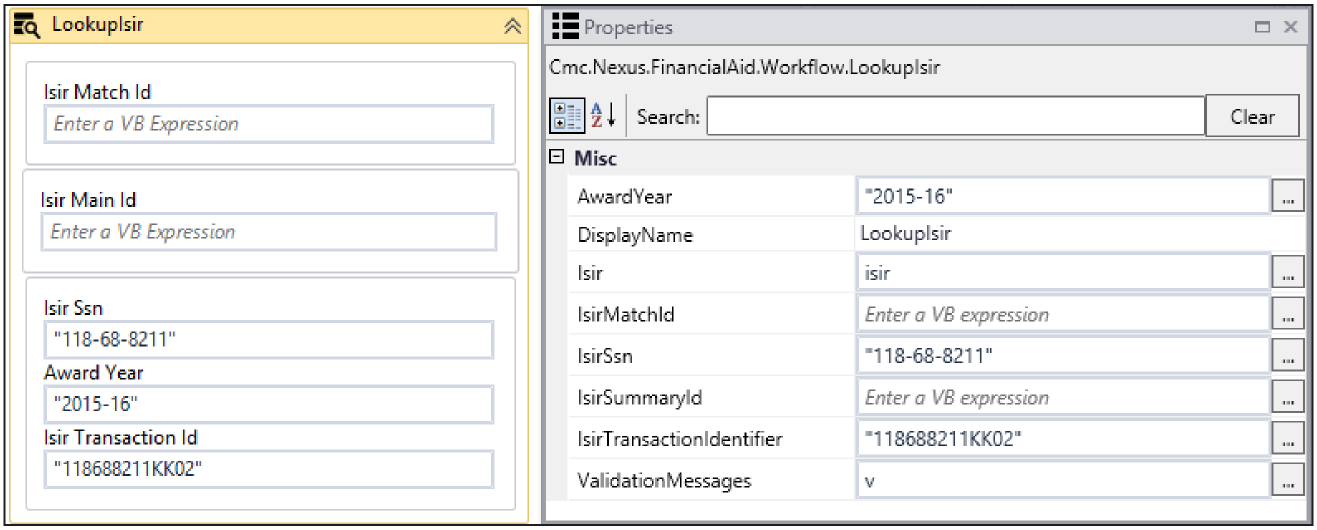
Award Year: Enter a VB Expression

Isir Transaction Id: Enter a VB Expression

Cmc.Nexus.FinancialAid.Workflow.LookupIsir

AwardYear	Enter a VB expression
DisplayName	LookupIsir
Isir	isir
IsirMatchId	Enter a VB expression
IsirSsn	Enter a VB expression
IsirSummaryId	2809
IsirTransactionIdentifier	Enter a VB expression
ValidationMessages	v

- LookupIsir based on **Award Year**, **SSN**, and **ISIR Transaction Id** (Conditionally required input values marked C3 in the table below)



Isir Match Id: Enter a VB Expression

Isir Main Id: Enter a VB Expression

Isir Ssn: "118-68-8211"

Award Year: "2015-16"

Isir Transaction Id: "118688211KK02"

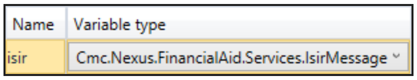
Cmc.Nexus.FinancialAid.Workflow.LookupIsir

AwardYear	"2015-16"
DisplayName	LookupIsir
Isir	isir
IsirMatchId	Enter a VB expression
IsirSsn	"118-68-8211"
IsirSummaryId	Enter a VB expression
IsirTransactionIdentifier	"118688211KK02"
ValidationMessages	v

Properties

LookupIsir Properties

Property	Value	Required	Notes
AwardYear	`InArgument<String>`	Conditional C3	Specify the Award Year using a string, for example, "2015-16".
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
Isir	`OutArgument<IsirMessage>`	Yes	The ISIR returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.FinancialAid.Contracts > Cmc.Nexus.FinancialAid.Services**, and select **IsirMessage**.  Name | Variable type isir | Cmc.Nexus.FinancialAid.Services.IsirMessage See IsirMessage Class in the Anthology Student Object Library.
IsirMatchId	`InArgument<Int32>`	Conditional C1	Specify the Id used to match ISIRs to Anthology Student Master records
IsirSsn	`InArgument<String>`	Conditional C3	Specify the SSN associated with ISIR records.
IsirSummaryId	`InArgument<Int32>`	Conditional C2	Specify the ISIR Main Id.
IsirTransactionIdentifier	`InArgument<String>`	Conditional C3	Specify the ISIR Transaction Id.
ValidationMessages	`InOutArgument<ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

UpdateISIRVerificationDependent

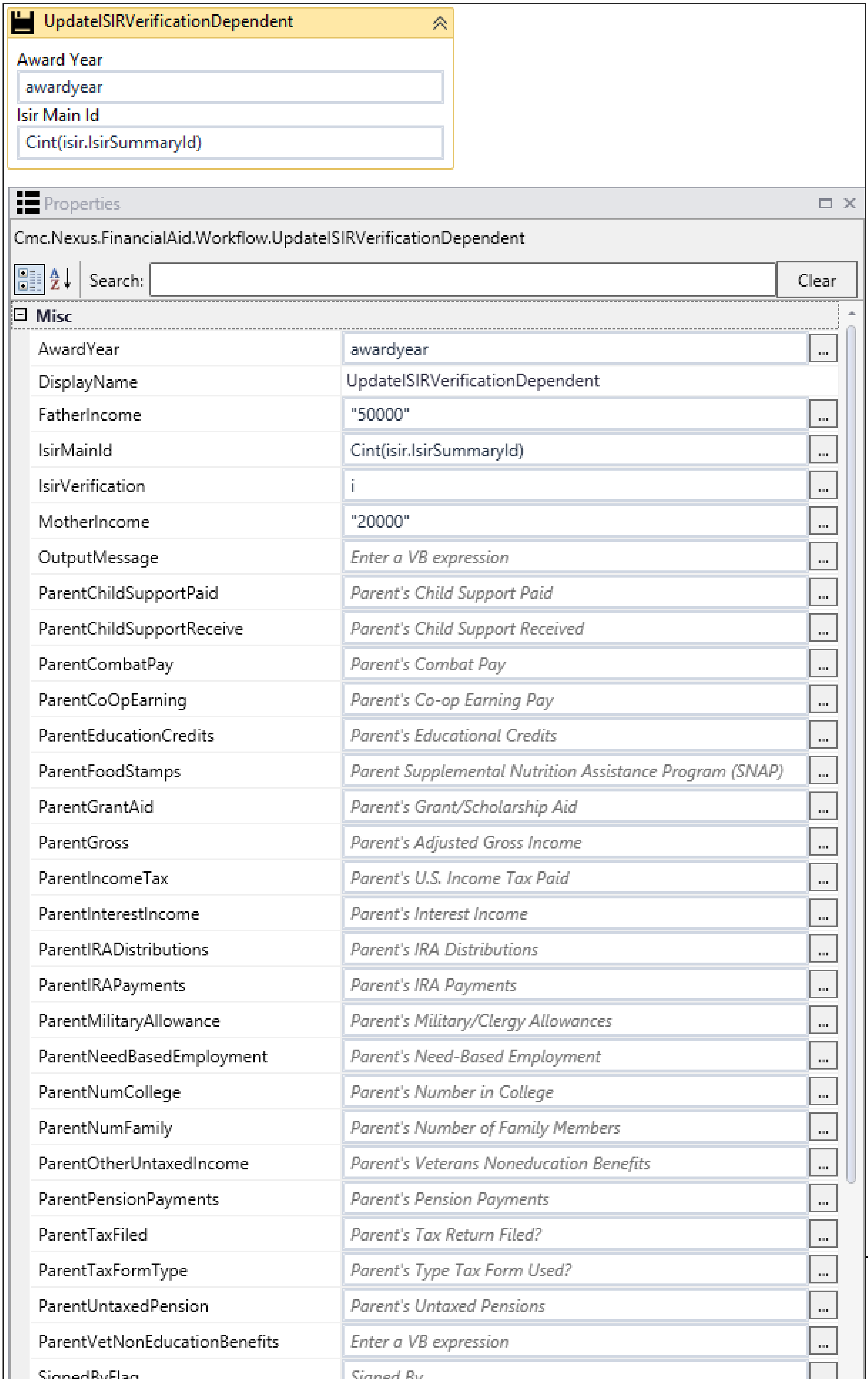
The UpdateISIRVerificationDependent activity is used to submit ISIR verification data for a dependent student. For a given award year, the activity captures the dependent student verification data.

The activity will only save the data submitted by the student or parent in Anthology Student. The Financial Aid staff at the institution will be responsible for performing the manual ISIR verification process.

When the UpdateISIRVerificationDependent activity is executed, the following occurs:

- The verification values are updated in Anthology Student. If corrections are pending, the update of values is not allowed.
- Field codes (SAR) are identified based on the award year schema.
- The values for each field are validated (compared with default values on the schema).
- After all validations have passed, a new record is written to the `faisirverification` table. If the record exists, the values are updated.
- The updated values are displayed on the ISIR verification form in Anthology Student.

See UpdateISIRVerificationDependent Example for an example of how this activity can be integrated in a workflow.



UpdateISIRVerificationDependent

Award Year: awardyear

Isir Main Id: Cint(isir.IsirSummaryId)

Cmc.Nexus.FinancialAid.Workflow.UpdateISIRVerificationDependent

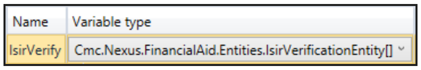
Misc

AwardYear	awardyear
DisplayName	UpdateISIRVerificationDependent
FatherIncome	"50000"
IsirMainId	Cint(isir.IsirSummaryId)
IsirVerification	i
MotherIncome	"20000"
OutputMessage	*Enter a VB expression*
ParentChildSupportPaid	*Parent's Child Support Paid*
ParentChildSupportReceive	*Parent's Child Support Received*
ParentCombatPay	*Parent's Combat Pay*
ParentCoOpEarning	*Parent's Co-op Earning Pay*
ParentEducationCredits	*Parent's Educational Credits*
ParentFoodStamps	*Parent Supplemental Nutrition Assistance Program (SNAP)*
ParentGrantAid	*Parent's Grant/Scholarship Aid*
ParentGross	*Parent's Adjusted Gross Income*
ParentIncomeTax	*Parent's U.S. Income Tax Paid*
ParentInterestIncome	*Parent's Interest Income*
ParentIRADistributions	*Parent's IRA Distributions*
ParentIRAPayments	*Parent's IRA Payments*
ParentMilitaryAllowance	*Parent's Military/Clergy Allowances*
ParentNeedBasedEmployment	*Parent's Need-Based Employment*
ParentNumCollege	*Parent's Number in College*
ParentNumFamily	*Parent's Number of Family Members*
ParentOtherUntaxedIncome	*Parent's Veterans Noneducation Benefits*
ParentPensionPayments	*Parent's Pension Payments*
ParentTaxFiled	*Parent's Tax Return Filed?*
ParentTaxFormType	*Parent's Type Tax Form Used?*
ParentUntaxedPension	*Parent's Untaxed Pensions*
ParentVetNonEducationBenefits	*Enter a VB expression*
SignedByFlag	*Signed By*

Properties

UpdateISIRVerificationDependent Properties

Property	Value	Required	Notes
AwardYear	`InArgument<String>`	Yes	Specify the Award Year using a VB expression or variable. Format: “XXXX-XX”. Example: “2015-16”.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
FatherIncome	`InArgument<String>`	No	Specify the Father's Income Earned From Work using a VB expression or variable. Valid field content: -9999999 to 9999999
IsirMainId	`InArgument<Int32>`	Yes	Specify the ISIR Main Id using a VB expression or variable.

Property	Value	Required	Notes
IsirVerification	`OutArgument <IsirVerificationEntity[]>`	Yes	The UpdateISIRVerificationIndependent activity returns an array of ISIR Verification values associated with the Award Year and ISIR Identification. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.FinancialAid.Entities**, select **IsirVerificationEntity**, and click **OK**.  Name | Variable type IsirVerify | Cmc.Nexus.FinancialAid.Entities.IsirVerificationEntity[] See IsirVerificationEntity Class in the Anthology Student Object Library.
MotherIncome	`InArgument<String>`	No	Specify the Mother's Income Earned From Work using a VB expression or variable. Valid field content: -9999999 to 9999999
OutputMessage	`OutArgument<String>`	No	Specify the Output Message using a VB expression or variable.
ParentChildSupportPaid	`InArgument<String>`	No	Specify the Parent's Child Support Paid using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
ParentChildSupportReceive	`InArgument<String>`	No	Specify the Parent's Child Support Received using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentCombatPay	`InArgument<String>`	No	Specify the Parent's Combat Pay using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentCoOpEarning	`InArgument<String>`	No	Specify the Parent's Co-op Earning Pay using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentEducationCredit	`InArgument<String>`	No	Specify the Parent's Educational Credits using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentFoodStamps	`InArgument<String>`	No	Specify whether the Parent Supplemental Nutrition Assistance Program (SNAP) applies. Valid field content: Yes or No
ParentGrantAid	`InArgument<String>`	No	Specify the Parent's Grant/Scholarship Aid using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentGross	`InArgument<String>`	No	Specify the Parent's Adjusted Gross Income using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentIncomeTax	`InArgument<String>`	No	Specify the Parent's U.S. Income Tax Paid using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
ParentInterestIncome	`InArgument<String>`	No	Specify the Parent's Interest Income using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentIRADistributions	`InArgument<String>`	No	Specify the Parent's IRA Distributions using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentIRAPayments	`InArgument<String>`	No	Specify the Parent's IRA Payments using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentMilitaryAllowance	`InArgument<String>`	No	Specify the Parent's Military/Clergy Allowances using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentNeedBased Employment	`InArgument<String>`	No	Specify the Parent's Need-Based Employment using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentNumCollege	`InArgument<String>`	No	Specify the Parent's Number in College using a VB expression or variable. Valid field content: 0 to 9
ParentNumFamily	`InArgument<String>`	No	Specify the Parent's Number of Family Members using a VB expression or variable. Valid field content: 00 to 99
ParentOtherUntaxedIncome	`InArgument<String>`	No	Specify the Parent's Other Untaxed Income using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
ParentPensionPayments	`InArgument<String>`	No	Specify the Parent's Pension Payments using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentTaxFiled	`InArgument<String>`	No	Specify the Parent's Tax Return Filed status using a VB expression or variable. Valid field content: 1, 2, or 3 Where: 1 = Already completed 2 = Will file 3 = Will not file
ParentTaxFormType	`InArgument<String>`	No	Specify the Parent's Type of Tax Form Used using a VB expression or variable. Valid field content: 1, 2, 3, or 4. Where: 1 = IRS 1040 2 = IRS 1040A or 1040 EZ 3 = Foreign tax return 4 = Tax return from Puerto Rico, a U.S. territory, or freely associated state
ParentUntaxedPension	`InArgument<String>`	No	Specify the Parent's Untaxed Pensions using a VB expression or variable. Valid field content: 0000000 to 9999999
ParentVetNonEducation Benefits	`InArgument<String>`	No	Specify the Parent's Veterans Noneducation Benefits using a VB expression or variable. Valid field content: 0000000 to 9999999
SignedByFlag	`InArgument<String>`	No	Specify the Signed By Flag using a VB expression or variable. Valid field content:
SpouseIncome	`InArgument<String>`	No	Specify the Spouse's Income Earned From Work using a VB expression or variable. Valid field content: -9999999 to 9999999

Property	Value	Required	Notes
StudChildSupportPaid	`InArgument<String>`	No	Specify the Student's Child Support Paid using a VB expression or variable Valid field content: 0000000 to 9999999
StudChildSupportReceive	`InArgument<String>`	No	Specify the Student's Child Support Received using a VB expression or variable. Valid field content: 0000000 to 9999999
StudCombatPay	`InArgument<String>`	No	Specify the Student's Combat Pay using a VB expression or variable. Valid field content: 0000000 to 9999999
StudCoOpEarning	`InArgument<String>`	No	Specify the Student's Co-op Earning Pay using a VB expression or variable. Valid field content: 0000000 to 9999999
StudEducationCredits	`InArgument<String>`	No	Specify the Student's Educational Credits using a VB expression or variable. Valid field content: 0000000 to 9999999
StudentFoodStamps	`InArgument<String>`	No	Specify whether the Student Supplemental Nutrition Assistance Program (SNAP) applies. Valid field content: 1 or 2 Where: 1 = Yes 2 = No
StudentGross	`InArgument<String>`	No	Specify the Student's Adjusted Gross Income using a VB expression or variable. Valid field content: 0000000 to 9999999
StudentIncome	`InArgument<String>`	No	Specify the Student's Income Earned From Work using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
StudentIncomeTax	`InArgument<String>`	No	Specify the Student's U.S. Income Tax Paid using a VB expression or variable. Valid field content: 0000000 to 9999999
StudentNumCollege	`InArgument<String>`	No	Specify the Student's Number in College using a VB expression or variable. Valid field content: 1 to 9
StudentNumFamily	`InArgument<String>`	No	Specify the Student's Number of Family Members using a VB expression or variable. Valid field content: 1 to 9
StudentTaxFiled	`InArgument<String>`	No	Specify the Student's Tax Return Filed status using a VB expression or variable. Valid field content: 1, 2, or 3 Where: 1 = Already completed 2 = Will file 3 = Will not file
StudentTaxFormType	`InArgument<String>`	No	Specify the Student's Type of 2014 Tax Form Used using a VB expression or variable Valid field content: 1, 2, 3, or 4. Where: 1 = IRS 1040 2 = IRS 1040A or 1040 EZ 3 = Foreign tax return 4 = Tax return from Puerto Rico, a U.S. territory, or freely associated state
StudentTaxFormType TaxReturnDate	`InArgument<String>`	No	Specify whether the Student Tax Return was Signed and Dated. Valid field content: Y or N Where: Y = Yes N = No **Note**: This property is required if the StudentTaxFormType property is populated.

Property	Value	Required	Notes
StudGrantAid	`InArgument<String>`	No	Specify the Student's Grant/Scholarship Aid using a VB expression or variable. Valid field content: 0000000 to 9999999
StudInterestIncome	`InArgument<String>`	No	Specify the Student's Interest Income using a VB expression or variable. Valid field content: 0000000 to 9999999
StudIraDistributions	`InArgument<String>`	No	Specify the Student's IRA Distributions using a VB expression or variable. Valid field content: 0000000 to 9999999
StudIraPayments	`InArgument<String>`	No	Specify the Student's IRA Payments using a VB expression or variable. Valid field content: 0000000 to 9999999
StudMilitaryAllowance	`InArgument<String>`	No	Specify the Student's Military/Clergy Allowances using a VB expression or variable. Valid field content: 0000000 to 9999999
StudNeedBased Employment	`InArgument<String>`	No	Specify the Student's Need-Based Employment using a VB expression or variable. Valid field content: 0000000 to 9999999
StudOtherNonReported MoneyReceived	`InArgument<String>`	No	Specify the Student's Other Non-Reported Money using a VB expression or variable. Valid field content: 0000000 to 9999999
StudOtherUntaxedIncome	`InArgument<String>`	No	Specify the Student's Other Untaxed Income using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
StudPensionPayments	`InArgument<String>`	No	Specify the Student's Pension Payments using a VB expression or variable. Valid field content: 0000000 to 9999999
StudUntaxedPension	`InArgument<String>`	No	Specify the Student's Untaxed Pensions using a VB expression or variable. Valid field content: 0000000 to 9999999
StudVetNonEducation Benefits	`InArgument<String>`	No	Specify the Student's Veterans Noneducation Benefits using a VB expression or variable. Valid field content: 0000000 to 9999999
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

UpdateISIRVerificationDependent Example

The UpdateISIRVerificationDependent activity can be used in a workflow sequence as follows:

1. Optional: Insert a **WriteLine** activity to mark the beginning of the sequence with a Text field similar to the following:

   ```
   "START THE WORKFLOW"
   ```

2. Insert an **Assign** activity to assign a value to a string variable named "studentid".

   The "studentid" value (e.g., "3400035") could be retrieved from a form created in Forms Builder.

3. Insert an **Assign** activity to assign a value to a string variable named "awardyear".

   The "awardyear" value (e.g., "2016-17") could be retrieved from a form created in Forms Builder.

4. Insert an **ExecuteDataReader** activity to query the database for the social security number associated with the student identifier.

   The CommandText for the query could be similar to the following:

   ```
   "SELECT SSN FROM SyStudent where SyStudentId=" + studentid
   ```

5. In the **ExecuteDataReader** activity, insert a Sequence that contains two Assign activities.

- In the first **Assign** activity, assign the value `CurrentRow("ssn").ToString()` to a string variable named "ssn".
- In the second **Assign** activity, assign the value `ssn.Replace("-", "")` to the "ssn" variable.

  This assignment replaces the "-" characters is the ssn value.

6. Insert an **Assign** activity that assigns an SQL statement to a string variable named "sqlGetTranId".

   The SQL statement could be similar to the following:

   ```
   "SELECT TOP 1 TransactionId FROM FaISIRAllvw WHERE CurrentSSN= " + ssn + " AND
   AwardYear='" + awardyear + "' ORDER BY TransactionId DESC"
   ```

7. Insert another **ExecuteDataReader** activity to retrieve a value for the transaction identifier.

   The CommandText could be similar to the following:

   ```
   sqlGetTranId
   ```

8. In the **ExecuteDataReader** activity, insert an **Assign** activity that assigns the value `CurrentRow("TransactionId").ToString()` to a string variable named "TransId".
9. Insert a **LookupIsir** activity. Specify a variable named "isir" as the out argument.

   Use the following input variables:

   - "awardyear" (assigned in step 3)
   - "ssn" (assigned in step 5)
   - "TranId" (assigned in step 8)

10. Insert the **UpdateISIRVerificationDependent** activity.

    - Specify a variable named "i" as the out argument.
    - Specify the variable "awardyear" in the AwardYear property.
    - Specify the following expression in the IsirMainId property: `Cint(isir.IsirSummaryId)`. This expression converts the ISIR Main Id to an integer.
    - Specify any other applicable properties.

11. Insert an **If** activity with the following Condition:

    ```
    i.Length > 0
    ```

    This condition checks if any IsirVerificationEntity values were returned by the UpdateISIRVerificationDependent activity.

12. In the **Then** branch of the If activity, insert a **ForEach** activity with the following properties:

    TypeArgument: `Cmc.Nexus.FinancialAid.Entities.IsirVerificationEntity`

    Values: `i`

13. In the **ForEach** activity, insert a **WriteLine** activity with the following text:

```
"OUTPUT: " + " ISIRMAIN ID: " + item.IsirSummaryId.ToString + " FIELD NUMBER: " +
item.FieldNumber + " NEW VALUE: " + item.NewValue
```

The following steps capture errors in the workflow.

14. Insert an **If** activity with the following Condition:

    `v.HasErrors`

    Where "v" is a variable of type ValidationMessageCollection.

15. In the **Then** branch of the If activity, insert a **ForEach** activity with the following properties:

    TypeArgument: `Cmc.Core.Eventing.ValidationMessage`

    Values: `v`

16. Add a sequence into the **ForEach** activity and insert the following activities into the sequence:

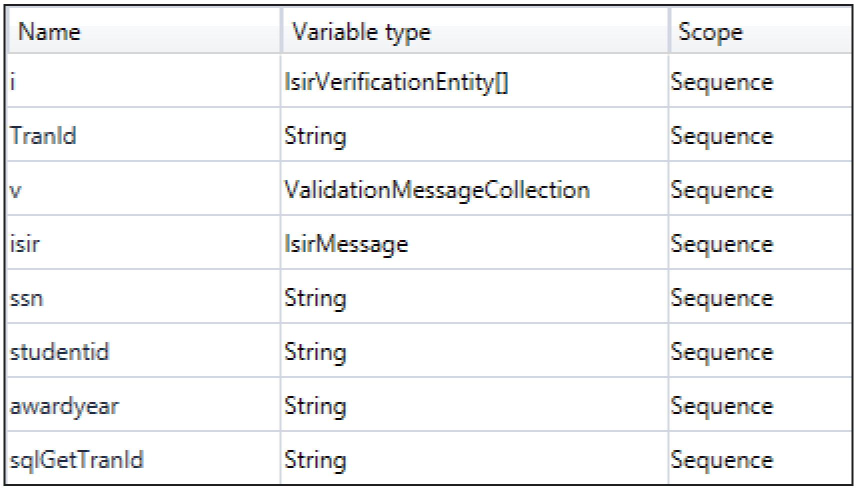
    - **LogLine**with the following Text: `Environment.NewLine &"ERROR VALIDATION: " & item.Message`

    - **WriteLine** with the following Text: `Environment.NewLine &"ERROR VALIDATION: " & item.Message`

17. In the **Else** branch of the If activity, insert a **WriteLine** activity with the text: `"NO ERROR"`

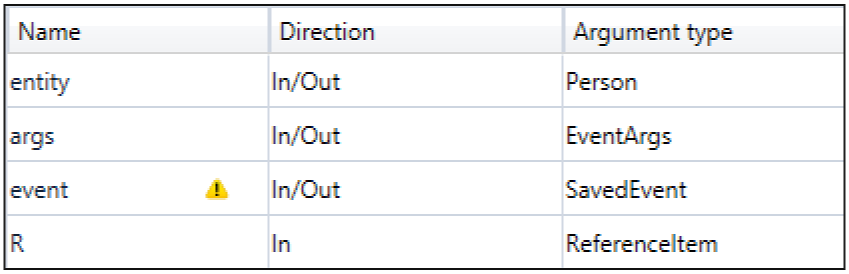
18. Optional: Add a final **WriteLine** activity to the workflow with the following text: `"END WORKFLOW"`

The following image summarizes the Variables used in the workflow:



Name	Variable type	Scope
i	IsirVerificationEntity[]	Sequence
TranId	String	Sequence
v	ValidationMessageCollection	Sequence
isir	IsirMessage	Sequence
ssn	String	Sequence
studentid	String	Sequence
awardyear	String	Sequence
sqlGetTranId	String	Sequence

The following image shows the Arguments associated with the workflow:



Name	Direction	Argument type
entity	In/Out	Person
args	In/Out	EventArgs
event ⚠	In/Out	SavedEvent
R	In	ReferenceItem

UpdateISIRVerificationIndependent

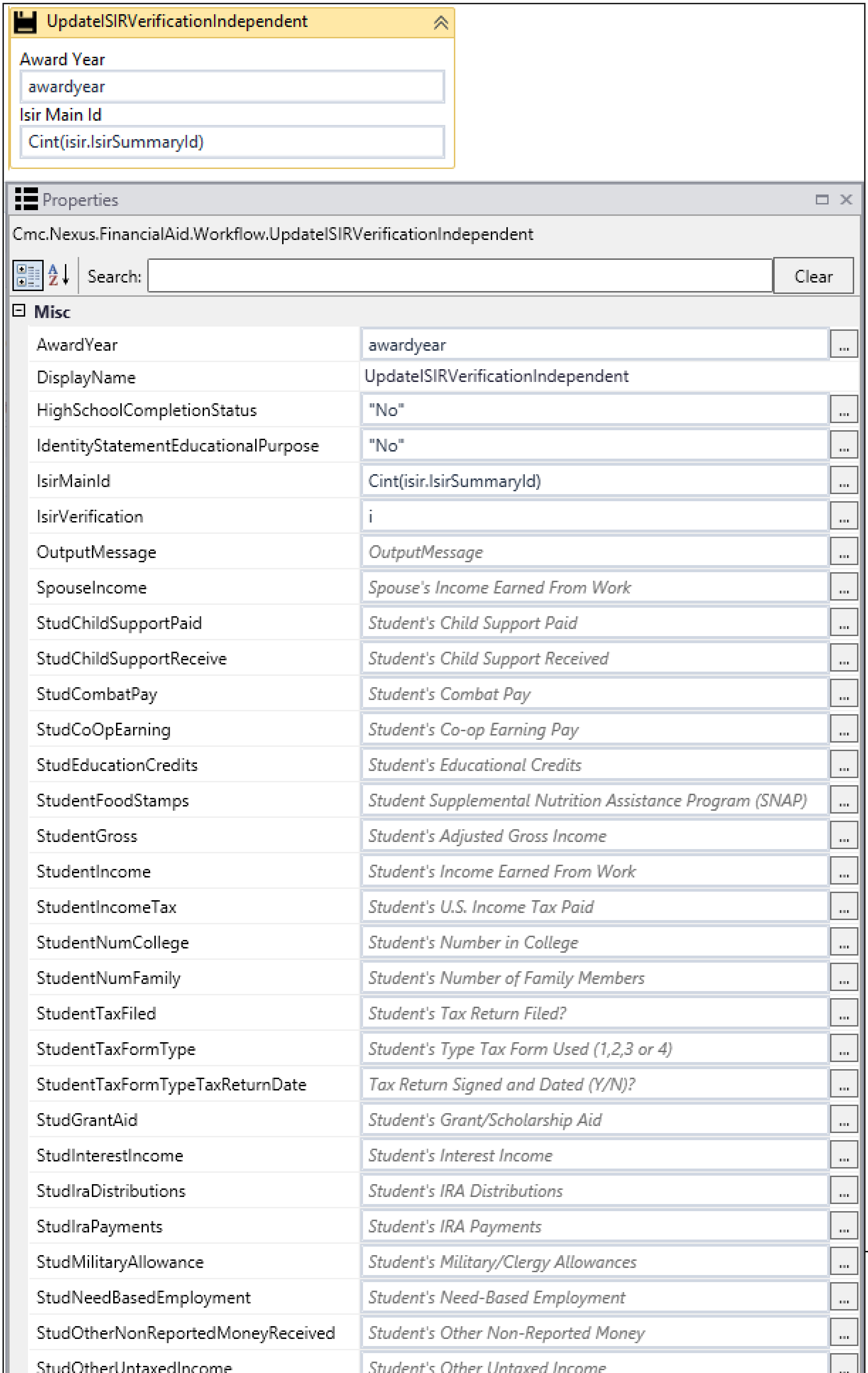
The UpdateISIRVerificationIndependent activity is used to submit ISIR verification data for an independent student. For a given award year, the activity captures the independent student verification data.

The activity will only save the data submitted by the student in Anthology Student. The Financial Aid staff at the institution will be responsible for performing the manual ISIR verification process.

When the UpdateISIRVerificationIndependent activity is executed, the following occurs:

- The verification values are updated in Anthology Student. If corrections are pending, the update of values is not allowed.
- Field codes (SAR) are identified based on the award year schema.
- The values for each field are validated (compared with default values on the schema).
- After all validations have passed, the `faisirverification` table is updated accordingly.
- The updated values are displayed on the ISIR verification form in Anthology Student.

See UpdateISIRVerificationIndependent Example for an example of how this activity can be integrated in a workflow.



UpdateISIRVerificationIndependent

Award Year
awardyear

Isir Main Id
Cint(isir.IsirSummaryId)

Properties

Cmc.Nexus.FinancialAid.Workflow.UpdateISIRVerificationIndependent

Search: Clear

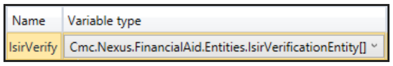
⊟ Misc

AwardYear	awardyear
DisplayName	UpdateISIRVerificationIndependent
HighSchoolCompletionStatus	"No"
IdentityStatementEducationalPurpose	"No"
IsirMainId	Cint(isir.IsirSummaryId)
IsirVerification	i
OutputMessage	*OutputMessage*
SpouseIncome	*Spouse's Income Earned From Work*
StudChildSupportPaid	*Student's Child Support Paid*
StudChildSupportReceive	*Student's Child Support Received*
StudCombatPay	*Student's Combat Pay*
StudCoOpEarning	*Student's Co-op Earning Pay*
StudEducationCredits	*Student's Educational Credits*
StudentFoodStamps	*Student Supplemental Nutrition Assistance Program (SNAP)*
StudentGross	*Student's Adjusted Gross Income*
StudentIncome	*Student's Income Earned From Work*
StudentIncomeTax	*Student's U.S. Income Tax Paid*
StudentNumCollege	*Student's Number in College*
StudentNumFamily	*Student's Number of Family Members*
StudentTaxFiled	*Student's Tax Return Filed?*
StudentTaxFormType	*Student's Type Tax Form Used (1,2,3 or 4)*
StudentTaxFormTypeTaxReturnDate	*Tax Return Signed and Dated (Y/N)?*
StudGrantAid	*Student's Grant/Scholarship Aid*
StudInterestIncome	*Student's Interest Income*
StudIraDistributions	*Student's IRA Distributions*
StudIraPayments	*Student's IRA Payments*
StudMilitaryAllowance	*Student's Military/Clergy Allowances*
StudNeedBasedEmployment	*Student's Need-Based Employment*
StudOtherNonReportedMoneyReceived	*Student's Other Non-Reported Money*
StudOtherUntaxedIncome	*Student's Other Untaxed Income*

Properties

UpdateISIRVerificationIndependent Properties

Property	Value	Required	Notes
AwardYear	`InArgument<String>`	Yes	Specify the Award Year using a VB expression or variable. Format: “XXXX-XX”. Example: “2015-16”.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
HighSchoolCompletionStatus	`InArgument<String>`	No	Specify the High School Completion Status using a VB expression or variable. Valid field content: “True” or “False” **Note**: Inserts or updates to this field are allowed only if the student belongs to Verification Group V4 or V5. Otherwise, the workflow returns the error *"Student does not belong to a valid Verification Group (V4 or V5).*”
IdentityStatement EducationalPurpose	`InArgument<String>`	No	Specify the Identity/Statement of Educational Purpose using a VB expression or variable. Valid field content: “True” or “False” **Note**: Inserts or updates to this field are allowed only if the student belongs to Verification Group V4 or V5. Otherwise, the workflow returns the error *"Student does not belong to a valid Verification Group (V4 or V5).*”
IsirMainId	`InArgument<Int32>`	Yes	Specify the ISIR Main Id using a VB expression or variable.

Property	Value	Required	Notes
IsirVerification	`OutArgument <IsirVerificationEntity[]>`	Yes	The UpdateISIRVerificationIndependent activity returns an array of ISIR Verification values associated with the Award Year and ISIR Identification. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Array of [T]**. In the 'Select Types' window, select **Browse for Types**, and click **OK**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.FinancialAid.Entities**, select **IsirVerificationEntity**, and click **OK**.  Name: IsirVerify Variable type: Cmc.Nexus.FinancialAid.Entities.IsirVerificationEntity[] See IsirVerificationEntity Class in the Anthology Student Object Library.
OutputMessage	`OutArgument<String>`	No	Specify the Output Message using a VB expression or variable.
SpouseIncome	`InArgument<String>`	No	Specify the Spouse's Income Earned From Work using a VB expression or variable. Valid field content: -9999999 to 9999999
StudChildSupportPaid	`InArgument<String>`	No	Specify the Student's Child Support Paid using a VB expression or variable Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
StudChildSupportReceive	`InArgument<String>`	No	Specify the Student's Child Support Received using a VB expression or variable. Valid field content: 0000000 to 9999999
StudCombatPay	`InArgument<String>`	No	Specify the Student's Combat Pay using a VB expression or variable. Valid field content: 0000000 to 9999999
StudCoOpEarning	`InArgument<String>`	No	Specify the Student's Co-op Earning Pay using a VB expression or variable. Valid field content: 0000000 to 9999999
StudEducationCredits	`InArgument<String>`	No	Specify the Student's Educational Credits using a VB expression or variable. Valid field content: 0000000 to 9999999
StudentFoodStamps	`InArgument<String>`	No	Specify whether the Student Supplemental Nutrition Assistance Program (SNAP) applies. Valid field content: 1 or 2 Where: 1 = Yes 2 = No
StudentGross	`InArgument<String>`	No	Specify the Student's Adjusted Gross Income using a VB expression or variable. Valid field content: 0000000 to 9999999
StudentIncome	`InArgument<String>`	No	Specify the Student's Income Earned From Work using a VB expression or variable. Valid field content: 0000000 to 9999999
StudentIncomeTax	`InArgument<String>`	No	Specify the Student's U.S. Income Tax Paid using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
StudentNumCollege	`InArgument<String>`	No	Specify the Student's Number in College using a VB expression or variable. Valid field content: 1 to 9
StudentNumFamily	`InArgument<String>`	No	Specify the Student's Number of Family Members using a VB expression or variable. Valid field content: 1 to 9
StudentTaxFiled	`InArgument<String>`	No	Specify the Student's Tax Return Filed status using a VB expression or variable. Valid field content: 1, 2, or 3 Where: 1 = Already completed 2 = Will file 3 = Will not file
StudentTaxFormType	`InArgument<String>`	No	Specify the Student's Type of 2014 Tax Form Used using a VB expression or variable. Valid field content: 1, 2, 3, or 4. Where: 1 = IRS 1040 2 = IRS 1040A or 1040 EZ 3 = Foreign tax return 4 = Tax return from Puerto Rico, a U.S. territory, or freely associated state
StudentTaxFormType TaxReturnDate	`InArgument<String>`	No	Specify whether the Student Tax Return was Signed and Dated. Valid field content: Y or N Where: Y = Yes N = No **Note**: This property is required if the StudentTaxFormType property is populated.
StudGrantAid	`InArgument<String>`	No	Specify the Student's Grant/Scholarship Aid using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
StudInterestIncome	`InArgument<String>`	No	Specify the Student's Interest Income using a VB expression or variable. Valid field content: 0000000 to 9999999
StudIraDistributions	`InArgument<String>`	No	Specify the Student's IRA Distributions using a VB expression or variable. Valid field content: 0000000 to 9999999
StudIraPayments	`InArgument<String>`	No	Specify the Student's IRA Payments using a VB expression or variable. Valid field content: 0000000 to 9999999
StudMilitaryAllowance	`InArgument<String>`	No	Specify the Student's Military/Clergy Allowances using a VB expression or variable. Valid field content: 0000000 to 9999999
StudNeedBasedEmployment	`InArgument<String>`	No	Specify the Student's Need-Based Employment using a VB expression or variable. Valid field content: 0000000 to 9999999
StudOtherNonReported MoneyReceived	`InArgument<String>`	No	Specify the Student's Other Non-Reported Money using a VB expression or variable. Valid field content: 0000000 to 9999999
StudOtherUntaxedIncome	`InArgument<String>`	No	Specify the Student's Other Untaxed Income using a VB expression or variable. Valid field content: 0000000 to 9999999
StudPensionPayments	`InArgument<String>`	No	Specify the Student's Pension Payments using a VB expression or variable. Valid field content: 0000000 to 9999999

Property	Value	Required	Notes
StudUntaxedPension	`InArgument<String>`	No	Specify the Student's Untaxed Pensions using a VB expression or variable. Valid field content: 0000000 to 9999999
StudVetNonEducationBenefits	`InArgument<String>`	No	Specify the Student's Veterans Noneducation Benefits using a VB expression or variable. Valid field content: 0000000 to 9999999
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

UpdateISIRVerificationIndependent Example

The UpdateISIRVerificationIndependent activity can be used in a workflow sequence as follows:

1. Optional: Insert a **WriteLine** activity to mark the beginning of the sequence with a Text field similar to the following:

   ```
   "START THE WORKFLOW"
   ```

2. Insert an **Assign** activity to assign a value to a string variable named "studentid".

   The "studentid" value (e.g., "3400035") could be retrieved from a form created in Forms Builder.

3. Insert an **Assign** activity to assign a value to a string variable named "awardyear".

   The "awardyear" value (e.g., "2016-17") could be retrieved from a form created in Forms Builder.

4. Insert an **ExecuteDataReader** activity to query the database for the social security number associated with the student identifier.

   The CommandText for the query could be similar to the following:

   ```
   "SELECT SSN FROM SyStudent where SyStudentId=" + studentid
   ```

5. In the **ExecuteDataReader** activity, insert a Sequence that contains two Assign activities.

   - In the first **Assign** activity, assign the value `CurrentRow("ssn").ToString()` to a string variable named "ssn".
   - In the second **Assign** activity, assign the value `ssn.Replace("-", "")` to the "ssn" variable.

     This assignment replaces the "-" characters is the ssn value.

6. Insert an **Assign** activity that assigns an SQL statement to a string variable named "sqlGetTranId".

The SQL statement could be similar to the following:

```
"SELECT TOP 1 TransactionId FROM FaISIRAllvw WHERE CurrentSSN= " + ssn + " AND
AwardYear='" + awardyear + "' ORDER BY TransactionId DESC"
```

7. Insert another **ExecuteDataReader** activity to retrieve a value for the transaction identifier.

   The CommandText could be similar to the following:

   ```
   sqlGetTranId
   ```

8. In the **ExecuteDataReader** activity, insert an **Assign** activity that assigns the value `CurrentRow("TransactionId").ToString()` to a string variable named "TransId".

9. Insert a **LookupIsir** activity. Specify a variable named "isir" as the out argument.

   Use the following input variables:

   - "awardyear" (assigned in step 3)
   - "ssn" (assigned in step 5)
   - "TranId" (assigned in step 8)

10. Insert the **UpdateISIRVerificationIndependent** activity.

    - Specify a variable named "i" as the out argument.
    - Specify the variable "awardyear" in the AwardYear property.
    - Specify the following expression in the IsirMainId property: `Cint(isir.IsirSummaryId)`. This expression converts the ISIR Main Id to an integer.
    - Specify any other applicable properties.

11. Insert an **If** activity with the following Condition:

    ```
    i.Length > 0
    ```

    This condition checks if any IsirVerificationEntity values were returned by the UpdateISIRVerificationIndependent activity.

12. In the **Then** branch of the If activity, insert a **ForEach** activity with the following properties:

    TypeArgument: `Cmc.Nexus.FinancialAid.Entities.IsirVerificationEntity`

    Values: `i`

13. In the **ForEach** activity, insert a **WriteLine** activity with the following text:

    ```
    "OUTPUT: " + " ISIRMAIN ID: " + item.IsirSummaryId.ToString + " FIELD NUMBER: " +
    item.FieldNumber + " NEW VALUE: " + item.NewValue
    ```

The following steps capture errors in the workflow.

14. Insert an **If** activity with the following Condition:

    ```
    v.HasErrors
    ```

    Where "v" is a variable of type ValidationMessageCollection.

15. In the **Then** branch of the If activity, insert a **ForEach** activity with the following properties:

    TypeArgument: `Cmc.Core.Eventing.ValidationMessage`

    Values: `v`

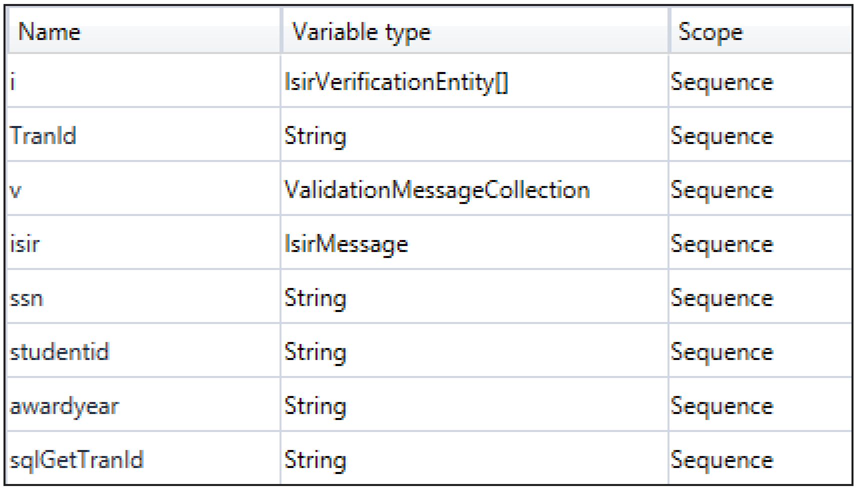
16. Add a sequence into the **ForEach** activity and insert the following activities into the sequence:

    - **LogLine**with the following Text: `Environment.NewLine &"ERROR VALIDATION: " & item.Message`
    - **WriteLine** with the following Text: `Environment.NewLine &"ERROR VALIDATION: " & item.Message`

17. In the **Else** branch of the If activity, insert a **WriteLine** activity with the text: `"NO ERROR"`

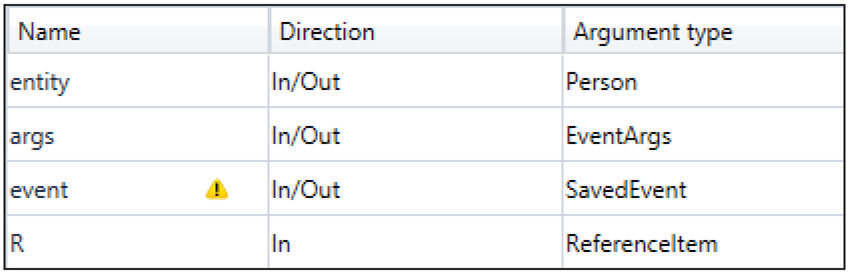
18. Optional: Add a final **WriteLine** activity to the workflow with the following text: `"END WORKFLOW"`

The following image summarizes the Variables used in the workflow:



Name	Variable type	Scope
i	IsirVerificationEntity[]	Sequence
TranId	String	Sequence
v	ValidationMessageCollection	Sequence
isir	IsirMessage	Sequence
ssn	String	Sequence
studentid	String	Sequence
awardyear	String	Sequence
sqlGetTranId	String	Sequence

The following image shows the Arguments associated with the workflow:



Name	Direction	Argument type
entity	In/Out	Person
args	In/Out	EventArgs
event ⚠	In/Out	SavedEvent
R	In	ReferenceItem

Cmc.Nexus.FormsBuilder.Workflow

Workflows created by Forms Builder version 3.x or later use workflow activities from the **Cmc.Nexus.FormsBuilder.Workflow** namespace. Please refer to Forms Builder help for information about these activities.

Cmc.Nexus.StudentAccounts.Workflow

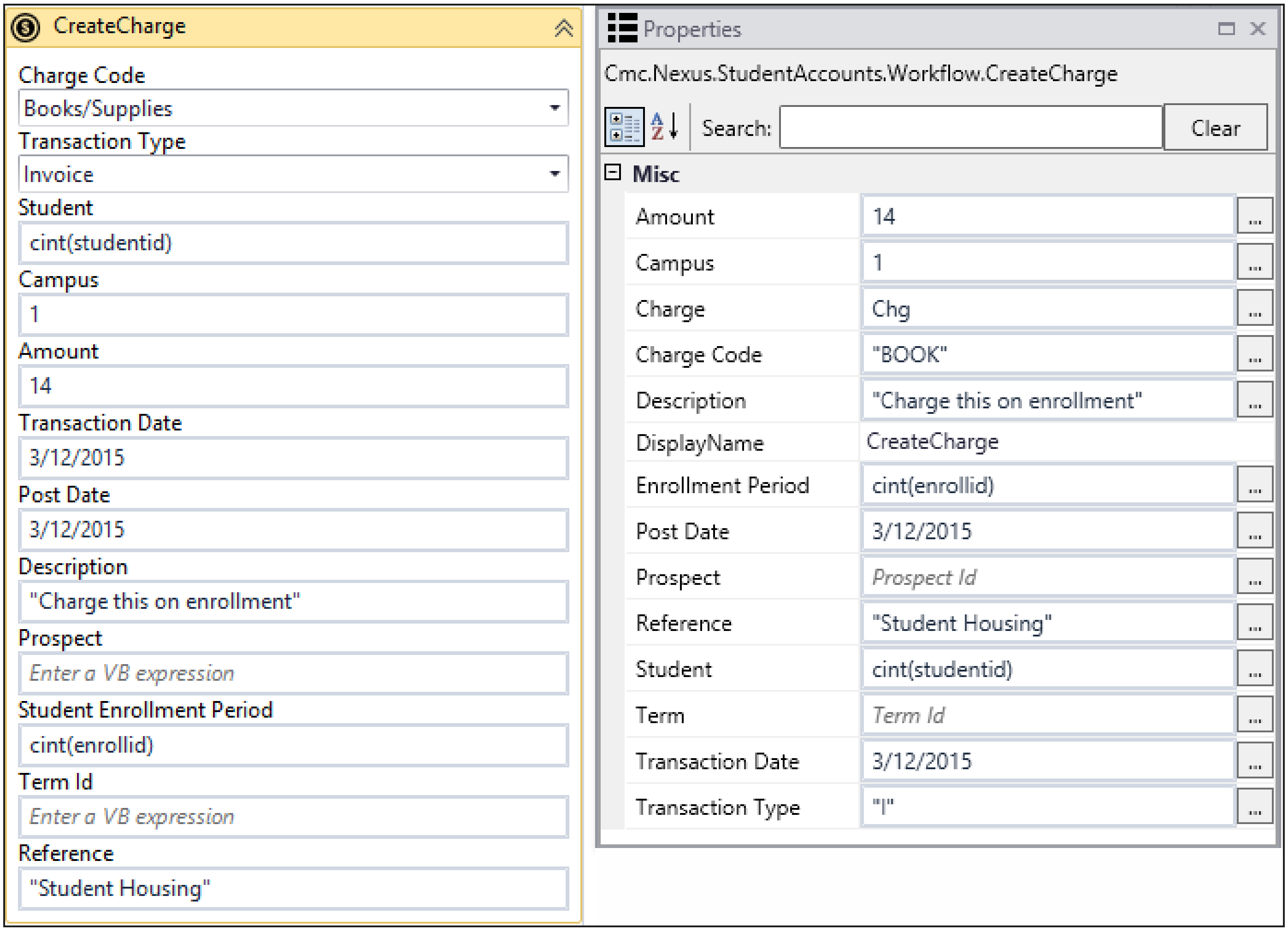
CreateCharge (V2)

Use the CreateCharge activity to post a Charge to an account associated with a Student Id.

The Charge (StudentAccountTransactionEntity) is the output of the workflow activity. You specify input properties such as Charge Code, Transaction Type, Student or Prospect Id, Transaction Date, Post Date, Description, Student Enrollment Period, and Reference.

You can use this activity to automate the process of posting charges when a condition that you create is true. A condition could be, for example, a change in enrollment, a grade change, or any other applicable event.

This activity creates an instance of a Charge; it does not save it to the database. To persist the Charge in the database, insert a SaveCharge (V2) activity.



CreateCharge

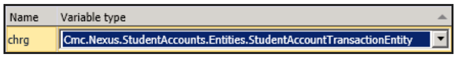
Charge Code	Books/Supplies
Transaction Type	Invoice
Student	cint(studentid)
Campus	1
Amount	14
Transaction Date	3/12/2015
Post Date	3/12/2015
Description	"Charge this on enrollment"
Prospect	*Enter a VB expression*
Student Enrollment Period	cint(enrollid)
Term Id	*Enter a VB expression*
Reference	"Student Housing"

Properties

Cmc.Nexus.StudentAccounts.Workflow.CreateCharge

Amount	14
Campus	1
Charge	Chg
Charge Code	"BOOK"
Description	"Charge this on enrollment"
DisplayName	CreateCharge
Enrollment Period	cint(enrollid)
Post Date	3/12/2015
Prospect	*Prospect Id*
Reference	"Student Housing"
Student	cint(studentid)
Term	*Term Id*
Transaction Date	3/12/2015
Transaction Type	"I"

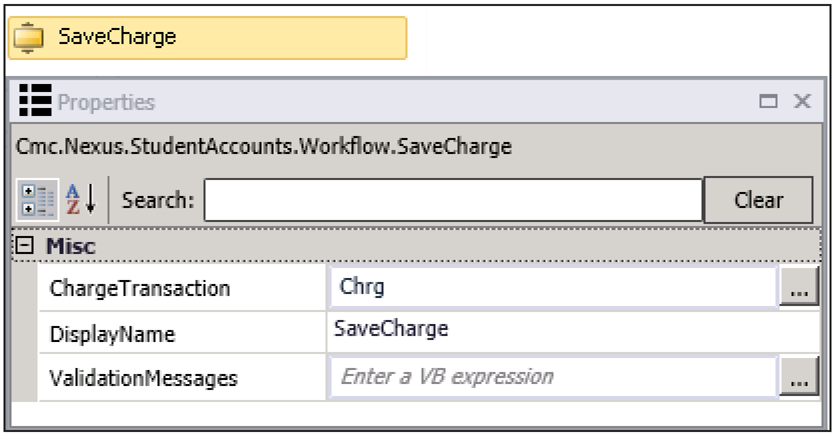
CreateCharge Properties

Property	Value	Required	Notes
Amount	`InArgument<Decimal>`	Yes	Specify the charge amount, for example, `98.50d`.
Campus	`InArgument<Int32>`	Yes	Specify the Campus Id using a VB expression or variable.
Charge	`OutArgument <StudentAccountTransactionEntity>`	Yes	The Charge that is posted to the account. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types....** In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentAccounts.Contracts > Cmc.Nexus.StudentAccounts.Entities**, select **StudentAccountTransactionEntity**, and click **OK**.  Name Variable type chrg Cmc.Nexus.StudentAccounts.Entities.StudentAccountTransactionEntity See StudentAccountTransactionEntity Class in the Anthology Student Object Library.
Charge Code	`InArgument<String>`	Yes	Select a value in the drop-down list of the activity in the Designer window.
Description	`InArgument<String>`	Yes	Specify a description of the Charge using a string, for example, `"Activity fee"`.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Enrollment Period	`InArgument<Int32>`	No	Specify the student enrollment period to which the Charge applies using a VB expression, for example, `entity.Id`.
Post Date	`InArgument<DateTime>`	Yes	Specify the date when the Charge is posted using a VB expression, for example, `DateTime.Now`.

Property	Value	Required	Notes
Prospect	`OutArgument<Int32>`	No	Specify the Prospect Id using a VB expression or variable, for example, `entity.Id`.
Reference	`InArgument<String>`	No	Specify a reference for the Charge using a string, for example, `"Engineering Lab"`.
Student	`InArgument<Int32>`	Yes	Specify the Student Id using a VB expression, for example, `entity.Id`.
Term	`InArgument<Nullable<Int32>>`	No	Specify the Term Id using a VB expression or variable.
Transaction Date	`InArgument<DateTime>`	Yes	Specify the transaction date using a VB expression, for example, `DateTime.Now`.
Transaction Type	`InArgument<String>`	Yes	Select a value in the drop-down list of the activity in the Designer window.

SaveCharge (V2)

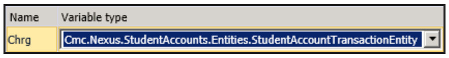
Use the SaveCharge activity to save a charge transaction and display a validation message.



Cmc.Nexus.StudentAccounts.Workflow.SaveCharge

ChargeTransaction	Chrg
DisplayName	SaveCharge
ValidationMessages	Enter a VB expression

SaveCharge Properties

Property	Value	Required	Notes
Charge Transaction	`InOutArgument <StudentAccountTransactionEntity>`	Yes	The Student Account charge transaction returned by the activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentAccounts.Contracts > Cmc.Nexus.StudentAccounts.Entities**, select **StudentAccountTransactionEntity**, and click **OK**.  Name / Variable type Chrg / Cmc.Nexus.StudentAccounts.Entities.StudentAccountTransactionEntity See StudentAccountTransactionEntity Class in the Anthology Student Object Library.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

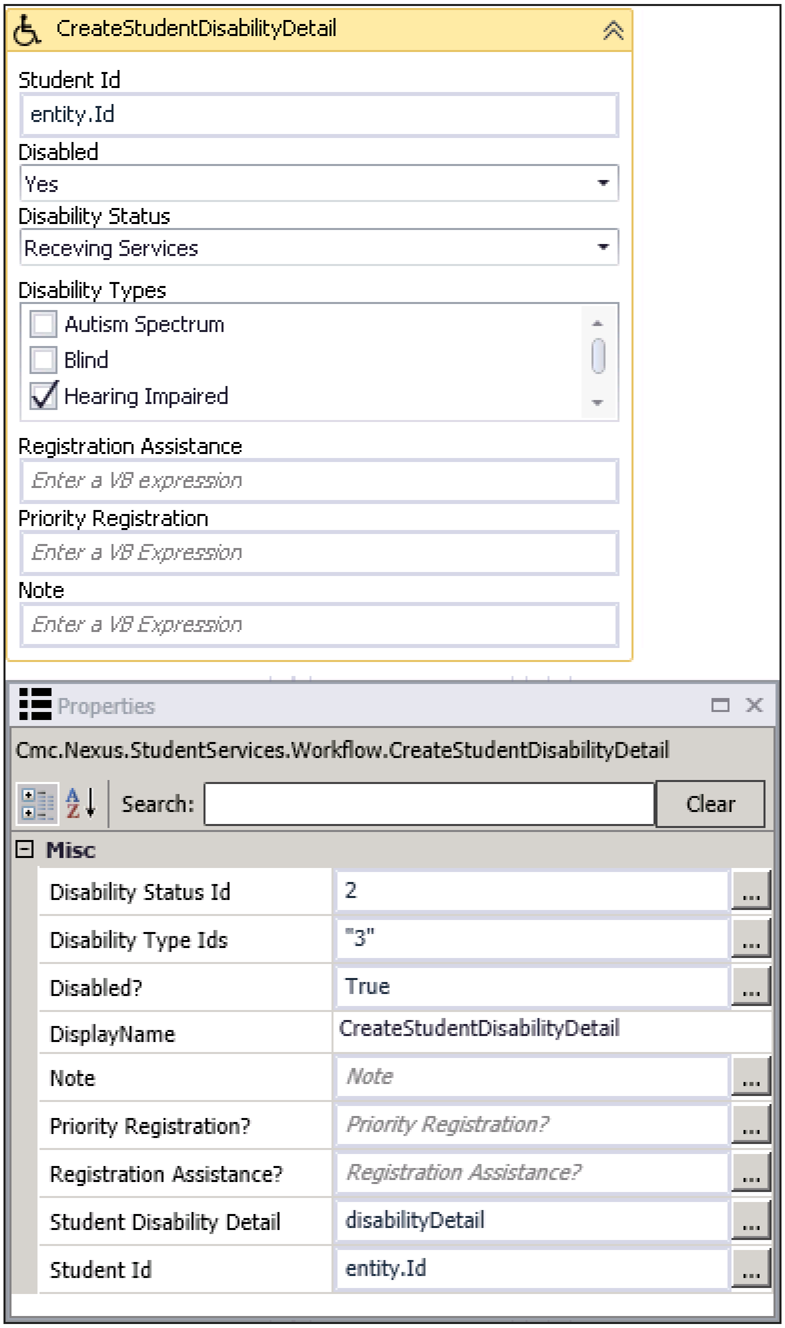
Cmc.Nexus.StudentServices.Workflow

CreateStudentDisabilityDetail (V2)

The CreateStudentDisabilityDetail activity creates an instance of a Student Disability Service record that can be passed to a SaveStudentDisabilityDetail (V2) activity.

Use Cases

- A workflow adds a disability service to a student enrolled in a term when the student selects a service available in Anthology Student from a Forms Builder form.
- A workflow adds a disability service record when the Disability Status is changed in the Student Master form in Anthology Student.



CreateStudentDisabilityDetail

Student Id: entity.Id

Disabled: Yes

Disability Status: Receving Services

Disability Types:
- ☐ Autism Spectrum
- ☐ Blind
- ☑ Hearing Impaired

Registration Assistance: Enter a VB expression

Priority Registration: Enter a VB Expression

Note: Enter a VB Expression

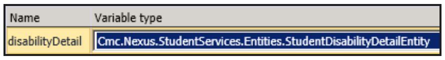
Cmc.Nexus.StudentServices.Workflow.CreateStudentDisabilityDetail

Disability Status Id	2
Disability Type Ids	"3"
Disabled?	True
DisplayName	CreateStudentDisabilityDetail
Note	Note
Priority Registration?	Priority Registration?
Registration Assistance?	Registration Assistance?
Student Disability Detail	disabilityDetail
Student Id	entity.Id

Properties

CreateStudentDisabilityDetail Properties

Property	Value	Required	Notes
Disability Status Id	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.

Property	Value	Required	Notes
Disability Type Ids	`InArgument<String>`	Yes	Select one or more values in the drop-down list of the activity in the Designer window.
Disabled?	`InArgument<Boolean>`	Yes	Select a value in the drop-down list of the activity in the Designer window. The default value is No.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Note	`InArgument<String>`	No	Specify a comment if applicable.
Priority Registration?	`InArgument<Nullable<Boolean>>`	No	A Boolean expression that specifies whether Priority Registration is required. The default value is null.
Registration Assistance?	`InArgument<Nullable<Boolean>>`	No	A Boolean expression that specifies whether Registration Assistance is required. The default value is null.
StudentDisabilityDetail	`OutArgument <StudentDisabilityDetailEntity>`	Yes	The Student Disability Detail value returned by the activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentService.Entities**, select **StudentDisabilityDetailEntity** and click **OK**.  Name Variable type disabilityDetail Cmc.Nexus.StudentServices.Entities.StudentDisabilityDetailEntity See StudentDisabilityDetailEntity Class in the Anthology Student Object Library.
Student Id	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.

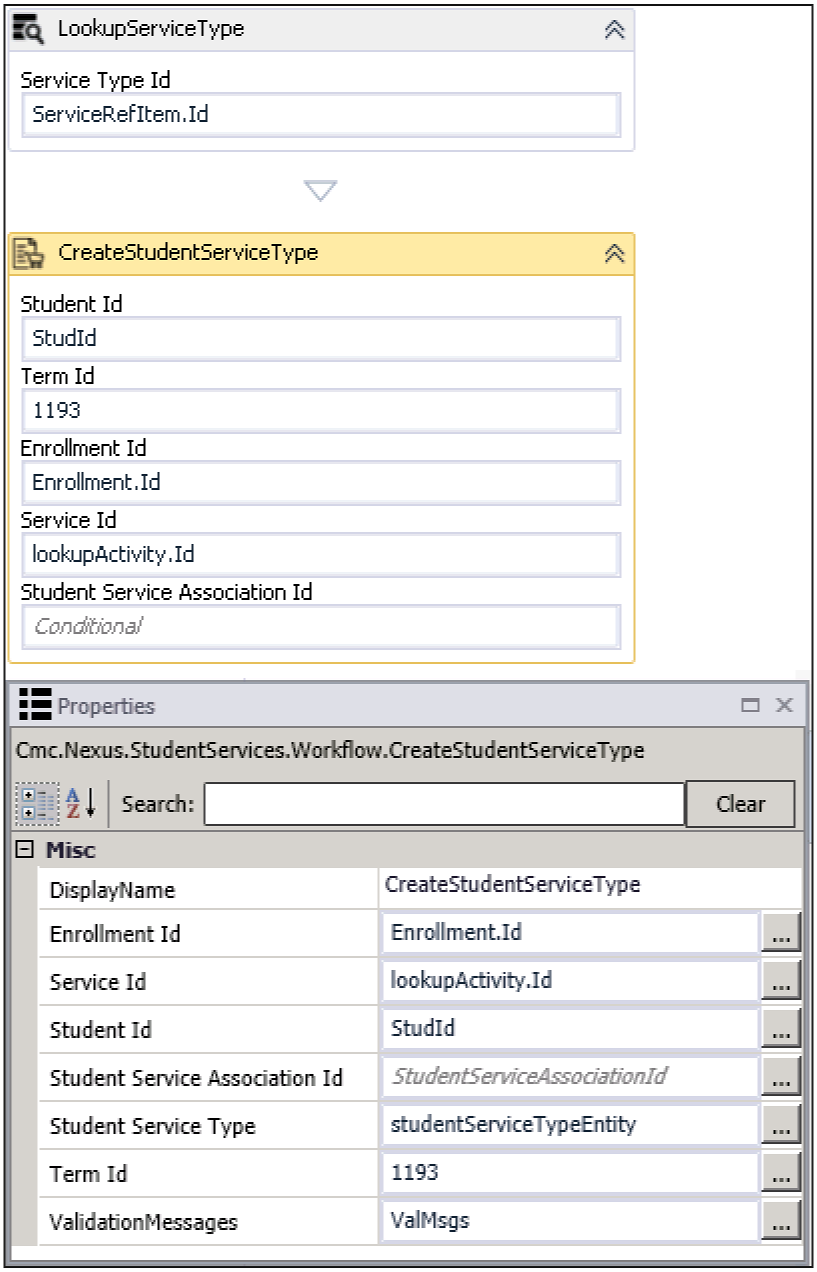
CreateStudentServiceType

Anthology Student enables users to configure non-academic, optionally billable, services for students. These services are associated with configurable service categories, for example, housing, meal plans, and so on. For billable school services that are not included in those provided by Anthology Student, users can add School-Defined Services and then create and associate Custom Fields with a Student Service. Anthology Student stores values entered in the Custom Fields on each instance of a service per student.

You can use the CreateStudentServiceType activity to create an instance of a Student Service Type record when a specific event occurs and pass it to a SaveStudentServiceType activity to persist the record in the database.

Example

From the Student Portal, a form sequence is created to add a meal plan. A student logs into the portal and clicks the link to add a meal plan. The first form verifies the student's current basic information (e.g., name, email). The student clicks Next, the form raises an event, and a workflow retrieves and displays the meal plan options. The student chooses a meal plan and the form raises another event. The workflow adds the service to the student record.



LookupServiceType

Service Type Id: ServiceRefItem.Id

CreateStudentServiceType

Student Id: StudId

Term Id: 1193

Enrollment Id: Enrollment.Id

Service Id: lookupActivity.Id

Student Service Association Id: *Conditional*

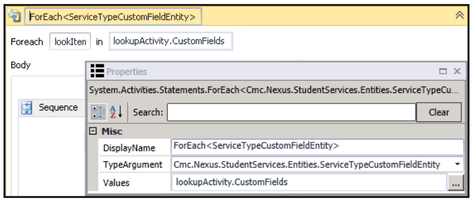
Cmc.Nexus.StudentServices.Workflow.CreateStudentServiceType

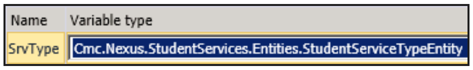
DisplayName	CreateStudentServiceType
Enrollment Id	Enrollment.Id
Service Id	lookupActivity.Id
Student Id	StudId
Student Service Association Id	*StudentServiceAssociationId*
Student Service Type	studentServiceTypeEntity
Term Id	1193
ValidationMessages	ValMsgs

Properties

CreateStudentServiceType Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Enrollment Id	`InArgument<Int32>`	Yes	Specify the Enrollment Id using a VB expression or variable.

Property	Value	Required	Notes
Service Id	`InArgument<Int32>`	Yes	Specify the Service Id using a VB expression or variable. **Note**: You can use a LookupServiceType activity to retrieve a Service Type Id.
Student Id	`InArgument<Int32>`	Yes	Specify the Student Id using a VB expression or variable.
Student Service Association Id	`InArgument<Int32>`	Conditional	If Custom Fields are defined for student services at your institution, specify the Student Service Association Id using a VB expression or variable. The values are stored in the table: SsStudentServiceCustomFieldAssociation. The entity name is ServiceTypeCustomFieldEntity. You can use a ForEach<> activity to capture the values of the ServiceTypeCustomFieldEntity.  ForEach<ServiceTypeCustomFieldEntity> Foreach lookIten in lookupActivity.CustomFields System.Activities.Statements.ForEach<Cmc.Nexus.StudentServices.Entities.ServiceTypeCu... DisplayName ForEach<ServiceTypeCustomFieldEntity> TypeArgument Cmc.Nexus.StudentServices.Entities.ServiceTypeCustomFieldEntity Values lookupActivity.CustomFields

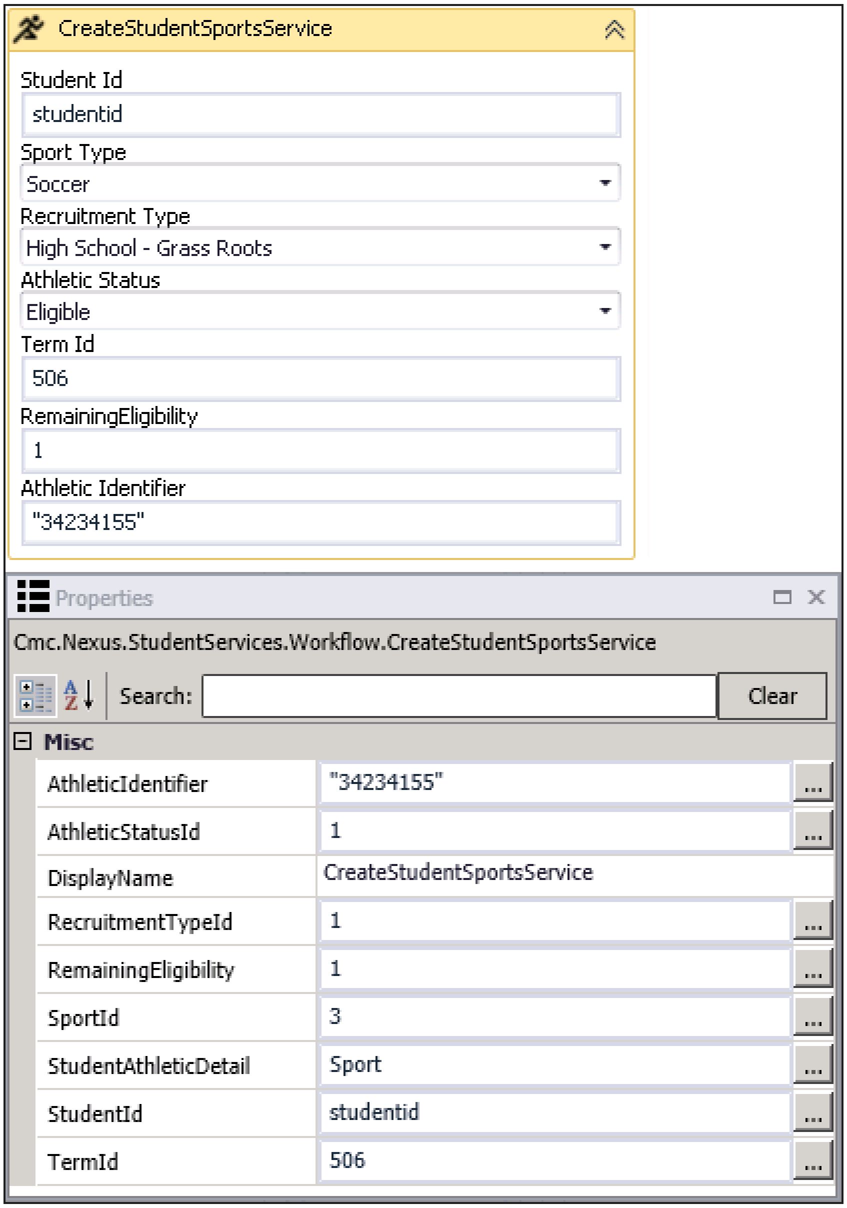
Property	Value	Required	Notes
Student Service Type	`OutArgument <StudentServiceTypeEntity>`	Yes	The Student Service Type value returned by the activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentService.Entities**, select **StudentServiceTypeEntity**, and click **OK**.  Name Variable type SrvType Cmc.Nexus.StudentServices.Entities.StudentServiceTypeEntity See StudentServiceTypeEntity Class in the Anthology Student Object Library.
Term Id	`InArgument<Int32>`	Yes	Specify the Term Id using a VB expression or variable.
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

CreateStudentSportsService (V2)

The CreateStudentSportsService activity creates an instance of a Student Sports Service record that can be passed to a SaveStudentSportsService (V2) activity.

Use Cases

- A workflow adds a sports service to a student enrolled in a term when the student selects a service available in Anthology Student from a Forms Builder form.
- A workflow adds a sports service record when a student is added to a sport group in Anthology Student.



CreateStudentSportsService

Student Id: studentid

Sport Type: Soccer

Recruitment Type: High School - Grass Roots

Athletic Status: Eligible

Term Id: 506

RemainingEligibility: 1

Athletic Identifier: "34234155"

Cmc.Nexus.StudentServices.Workflow.CreateStudentSportsService

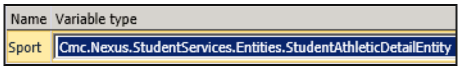
Misc

AthleticIdentifier	"34234155"
AthleticStatusId	1
DisplayName	CreateStudentSportsService
RecruitmentTypeId	1
RemainingEligibility	1
SportId	3
StudentAthleticDetail	Sport
StudentId	studentid
TermId	506

Properties

CreateStudentSportsService Properties

Property	Value	Required	Notes
AthleticIdentifier	`InArgument<String>`	No *****	Specify the Athletic Identifier using a VB expression or variable. If the Athletic Identifier is not supplied, the CreateStudentSportsService activity will look up if one exists in the SyStudent table for that student. ***** The Athletic Identifier is required if it has not already been defined.
AthleticStatusId	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
RecruitmentTypeId	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.
RemainingEligibility	`InArgument<Int32>`	Yes	Specify the Remaining Eligibility using a VB expression or variable.
SportId	`InArgument<Int32>`	Yes	Select a value in the drop-down list of the activity in the Designer window.

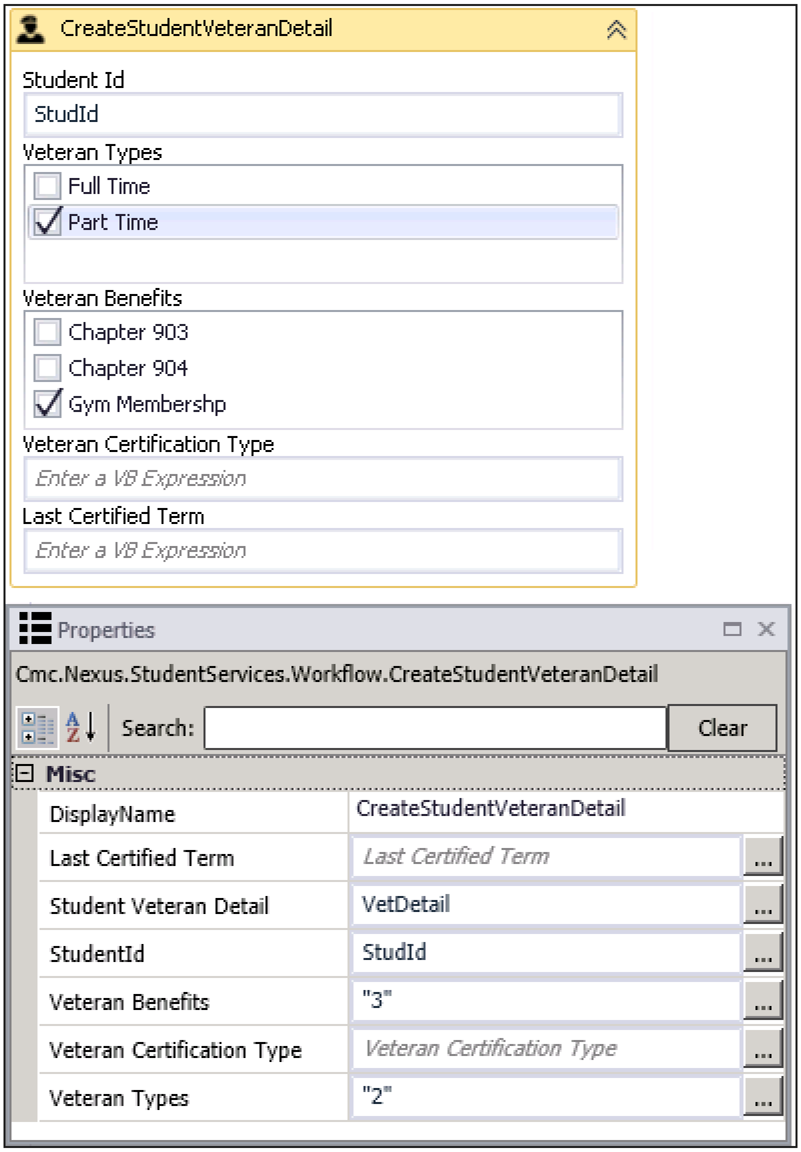
Property	Value	Required	Notes
StudentAthleticDetail	`OutArgument <StudentAthleticDetailEntity>`	Yes	The Student Athletic Detail value returned by the activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types....** In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentServices.Entities**, select **StudentAthleticDetailEntity** and click **OK**.  Name Variable type Sport Cmc.Nexus.StudentServices.Entities.StudentAthleticDetailEntity See StudentAthleticDetailEntity Class in the Anthology Student Object Library.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
TermId	`InArgument<Int32>`	Yes	Specify the Term Id using a VB expression or variable.

CreateStudentVeteranDetail (V2)

The CreateStudentVeteranDetail activity creates an instance of a Student Veteran Service record that can be passed to a SaveStudentVeteranDetail (V2) activity.

Use Cases

- A workflow adds a veteran service to a student enrolled in a term when the student selects a service available in Anthology Student from a Forms Builder form.
- A workflow adds a veteran service record when the Veteran Status is changed in the Student Master form in Anthology Student.



CreateStudentVeteranDetail

Student Id: StudId

Veteran Types

- □ Full Time
- ☑ Part Time

Veteran Benefits

- □ Chapter 903
- □ Chapter 904
- ☑ Gym Membershp

Veteran Certification Type: *Enter a VB Expression*

Last Certified Term: *Enter a VB Expression*

Cmc.Nexus.StudentServices.Workflow.CreateStudentVeteranDetail

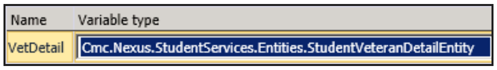
Misc

DisplayName	CreateStudentVeteranDetail
Last Certified Term	*Last Certified Term*
Student Veteran Detail	VetDetail
StudentId	StudId
Veteran Benefits	"3"
Veteran Certification Type	*Veteran Certification Type*
Veteran Types	"2"

Properties

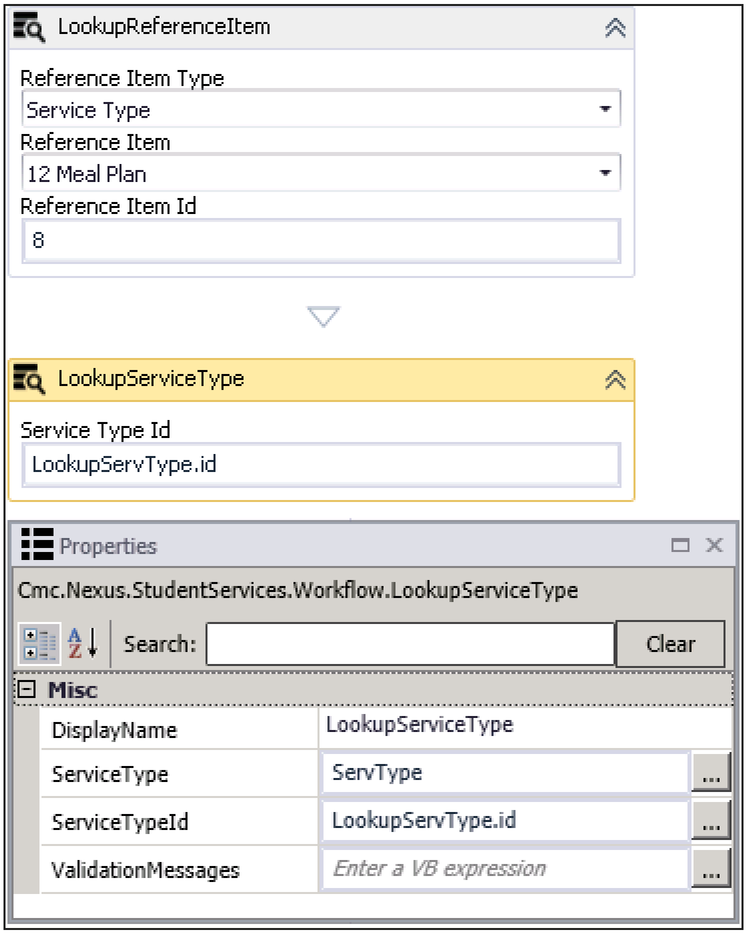
CreateStudentVeteranDetail Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Last Certified Term	`InArgument<Nullable<Int32>>`	No	A Boolean expression that specifies the Last Certified Term. The default value is null.

Property	Value	Required	Notes
Student Veteran Detail	`OutArgument <StudentVeteranDetailEntity>`	Yes	The Student Veteran Detail value returned by the activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentServices.Entities**, select **StudentVeteranDetailEntity** and click OK.  Name — Variable type VetDetail — Cmc.Nexus.StudentServices.Entities.StudentVeteranDetailEntity See StudentVeteranDetailEntity Class in the Anthology Student Object Library.
StudentId	`InArgument<Int32>`	Yes	Specify a Student Id using a VB expression or variable.
Veteran Benefits	`InArgument<Int32>`	Yes	Select one or more values in the drop-down list of the activity in the Designer window.
Veteran Certification Type	`InArgument<Nullable<Int32>>`	No	A Boolean expression that specifies a Veteran Certification Type. The default value is null.
Veteran Types	`InArgument<String>`	Yes	Select one or more values in the drop-down list of the activity in the Designer window.

LookupServiceType

The LookupServiceType activity is a lookup function that returns the ServiceTypeEntity from the SsService table within the Anthology Student database. Examples of student service types are parking passes, private tutoring, season tickets to a sporting event, meal plans, and so on. The ServiceTypeEntity can be used as input for the CreateStudentServiceType activity.



LookupReferenceItem

Reference Item Type: Service Type

Reference Item: 12 Meal Plan

Reference Item Id: 8

LookupServiceType

Service Type Id: LookupServType.id

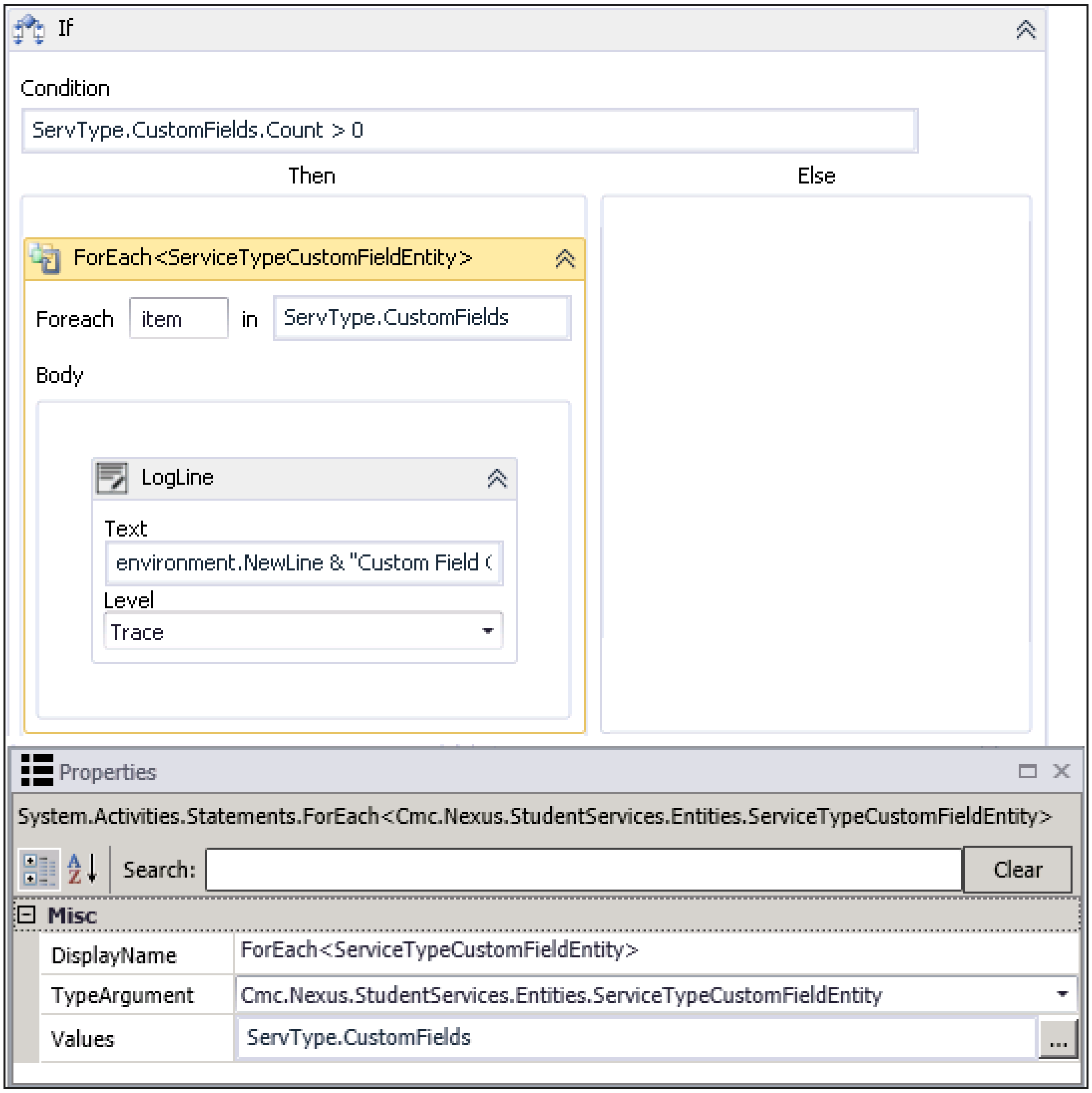
Cmc.Nexus.StudentServices.Workflow.LookupServiceType

Misc

DisplayName	LookupServiceType
ServiceType	ServType
ServiceTypeId	LookupServType.id
ValidationMessages	*Enter a VB expression*

If CustomFields exist for the ServiceTypeEntity, the LookupServiceType activity can return the values of the CustomFields collection from the SsStudentServiceCustomField table in Anthology Student database. See ServiceTypeCustomFieldsEntity Class in the Anthology Student Object Library.

In the example below, the condition `ServType.CustomFields.Count > 0` checks for CustomFields. If Custom Fields are found, the ForEach<> activity checks each field in the ServiceTypeCustomFieldsEntity. The subsequent LogLine activity captures the values of the CustomFields collection.



Condition: ServType.CustomFields.Count > 0

ForEach<ServiceTypeCustomFieldEntity>

Foreach item in ServType.CustomFields

Text: environment.NewLine & "Custom Field (

Level: Trace

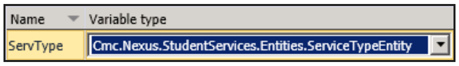
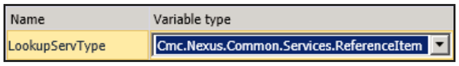
System.Activities.Statements.ForEach<Cmc.Nexus.StudentServices.Entities.ServiceTypeCustomFieldEntity>

DisplayName	ForEach<ServiceTypeCustomFieldEntity>
TypeArgument	Cmc.Nexus.StudentServices.Entities.ServiceTypeCustomFieldEntity
Values	ServType.CustomFields

Properties

LookupServiceType Properties

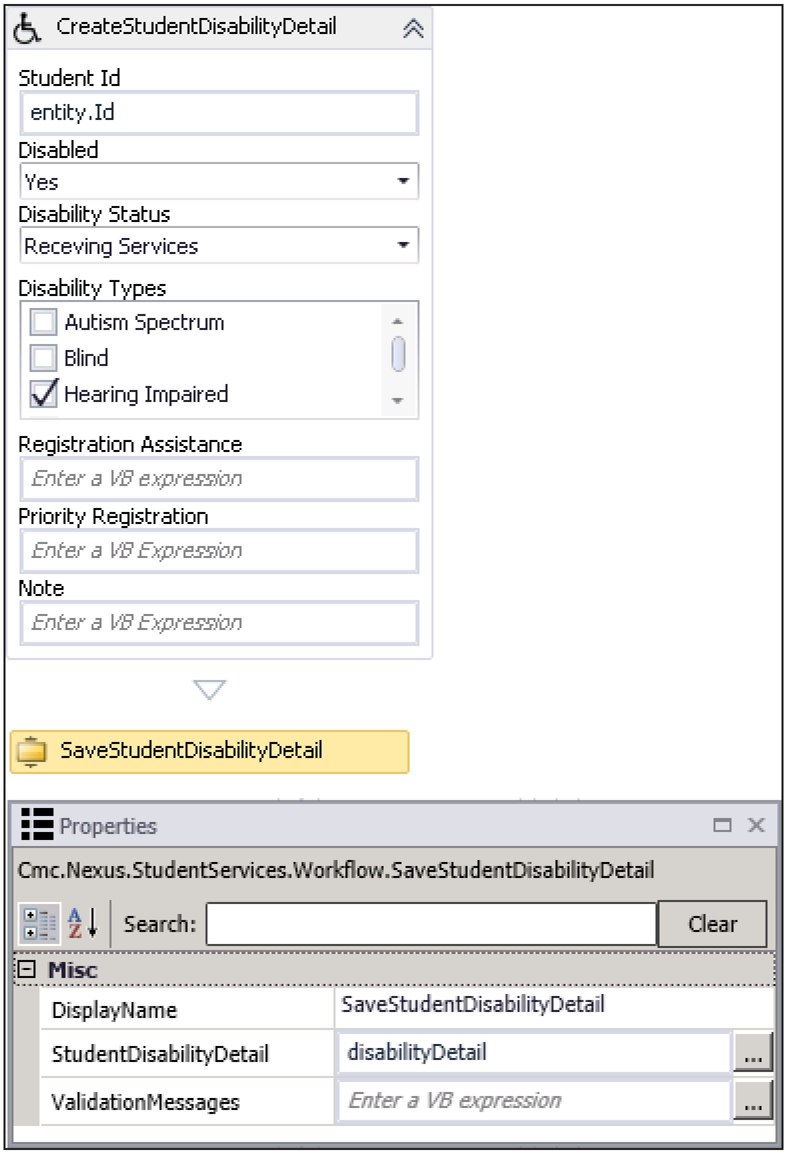
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
ServiceType	`OutArgument<ServiceTypeEntity>`	Yes	The ServiceTypeEntity returned by the lookup function. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types....** In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentServices.Entities**, select **ServiceTypeEntity** and click **OK**.  Name: ServType Variable type: Cmc.Nexus.StudentServices.Entities.ServiceTypeEntity See ServiceTypeEntity Class in the Anthology Student Object Library.
ServiceTypeId	`InArgument<Int32>`	No	The ServiceTypeId captured from an event. In this example above, the ServiceTypeId is obtained using a LookupReferenceItem activity with a Reference Item Type selection of "Service Type". The variable "LookupServType" is assigned to the LookupReferenceItem OutArgument. The Id associated the LookupServType variable is used as input for LookupServiceType.  Name: LookupServType Variable type: Cmc.Nexus.Common.Services.ReferenceItem
ValidationMessages	`OutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

SaveStudentDisabilityDetail (V2)

The SaveStudentDisabilityDetail activity saves a Student Disability Detail record that was created with the CreateStudentDisabilityDetail (V2) activity.

Note: If a record exists in the SsStudentDisabilityDetail table for the StudentId supplied in the CreateStudentDisabilityDetail activity, the SaveStudentDisabilityDetail activity updates the student's record.



CreateStudentDisabilityDetail

Student Id: entity.Id

Disabled: Yes

Disability Status: Receving Services

Disability Types:
- □ Autism Spectrum
- □ Blind
- ☑ Hearing Impaired

Registration Assistance: Enter a VB expression

Priority Registration: Enter a VB Expression

Note: Enter a VB Expression

Cmc.Nexus.StudentServices.Workflow.SaveStudentDisabilityDetail

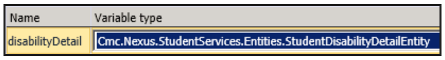
Misc

DisplayName	SaveStudentDisabilityDetail
StudentDisabilityDetail	disabilityDetail
ValidationMessages	Enter a VB expression

Properties

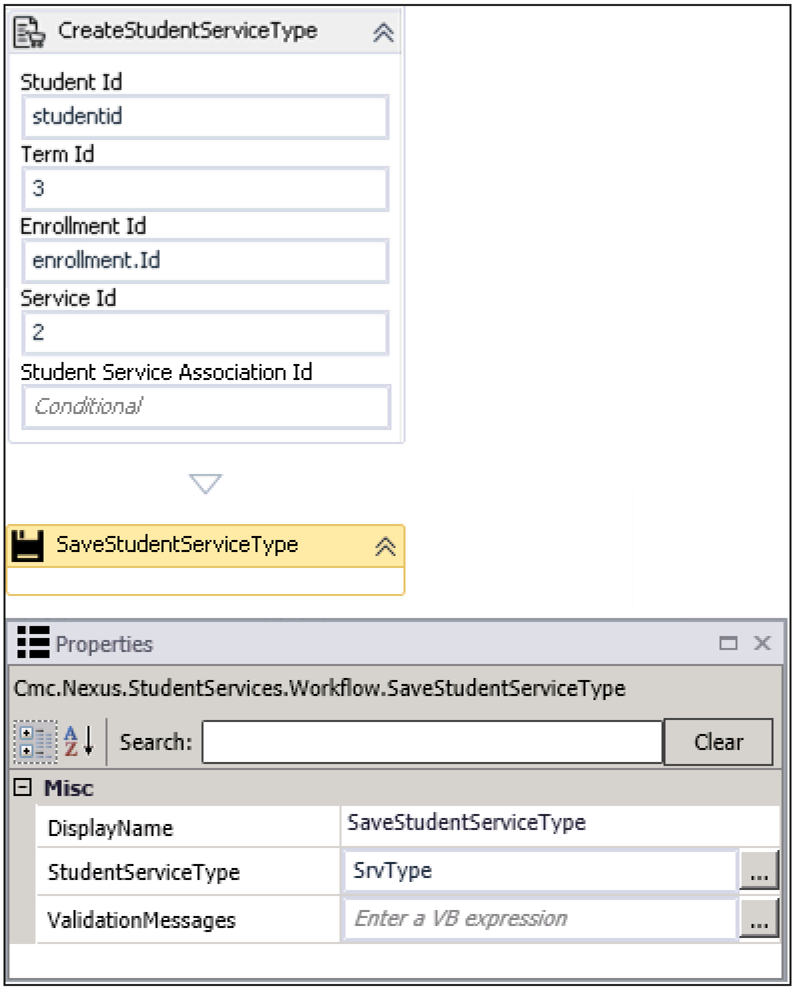
SaveStudentDisabilityDetail Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
StudentDisabilityDetail	`InOutArgument <StudentDisabilityDetailEntity>`	Yes	Specify the StudentDisabilityDetail entity to be saved using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentService.Entities**, select **StudentDisabilityDetailEntity** and click **OK**.  Name: disabilityDetail Variable type: Cmc.Nexus.StudentServices.Entities.StudentDisabilityDetailEntity See StudentDisabilityDetailEntity Class in the Anthology Student Object Library.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

SaveStudentServiceType

The SaveStudentServiceType activity saves a Student Service Type record that was created with the CreateStudentServiceType activity.



CreateStudentServiceType

Student Id: studentid

Term Id: 3

Enrollment Id: enrollment.Id

Service Id: 2

Student Service Association Id: *Conditional*

SaveStudentServiceType

Properties

Cmc.Nexus.StudentServices.Workflow.SaveStudentServiceType

Search:

Misc

DisplayName: SaveStudentServiceType

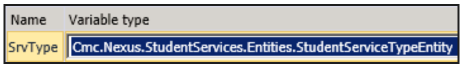
StudentServiceType: SrvType

ValidationMessages: *Enter a VB expression*

Properties

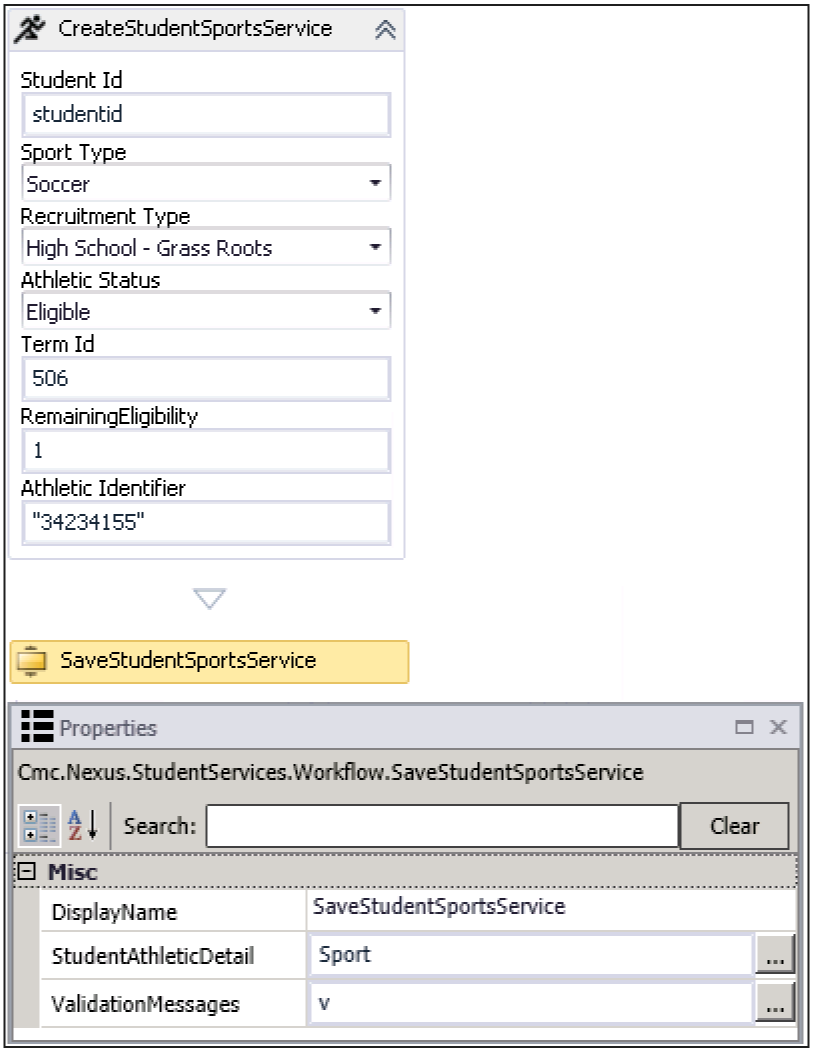
SaveStudentServiceType Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
StudentServiceType	`InOutArgument <StudentServiceTypeEntity>`	Yes	The Student Service Type value returned by the activity. This is a variable that can be used as input for subsequent activities in the workflow. Specify the variable's name, type, and scope (and default if applicable) in the Variables pane of the Designer window. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentService.Entities**, select **StudentServiceTypeEntit** and click **OK**.  Name Variable type SrvType Cmc.Nexus.StudentServices.Entities.StudentServiceTypeEntity See StudentServiceTypeEntity Class in the Anthology Student Object Library.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see [Capture Validation Errors]().

SaveStudentSportsService (V2)

The SaveStudentSportsService activity saves a Student Sports Service record that was created with the [CreateStudentSportsService (V2)]() activity.



CreateStudentSportsService

Student Id: studentid

Sport Type: Soccer

Recruitment Type: High School - Grass Roots

Athletic Status: Eligible

Term Id: 506

RemainingEligibility: 1

Athletic Identifier: "34234155"

SaveStudentSportsService

Properties

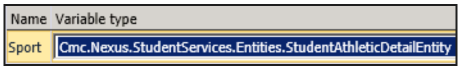
Cmc.Nexus.StudentServices.Workflow.SaveStudentSportsService

DisplayName	SaveStudentSportsService
StudentAthleticDetail	Sport
ValidationMessages	v

Properties

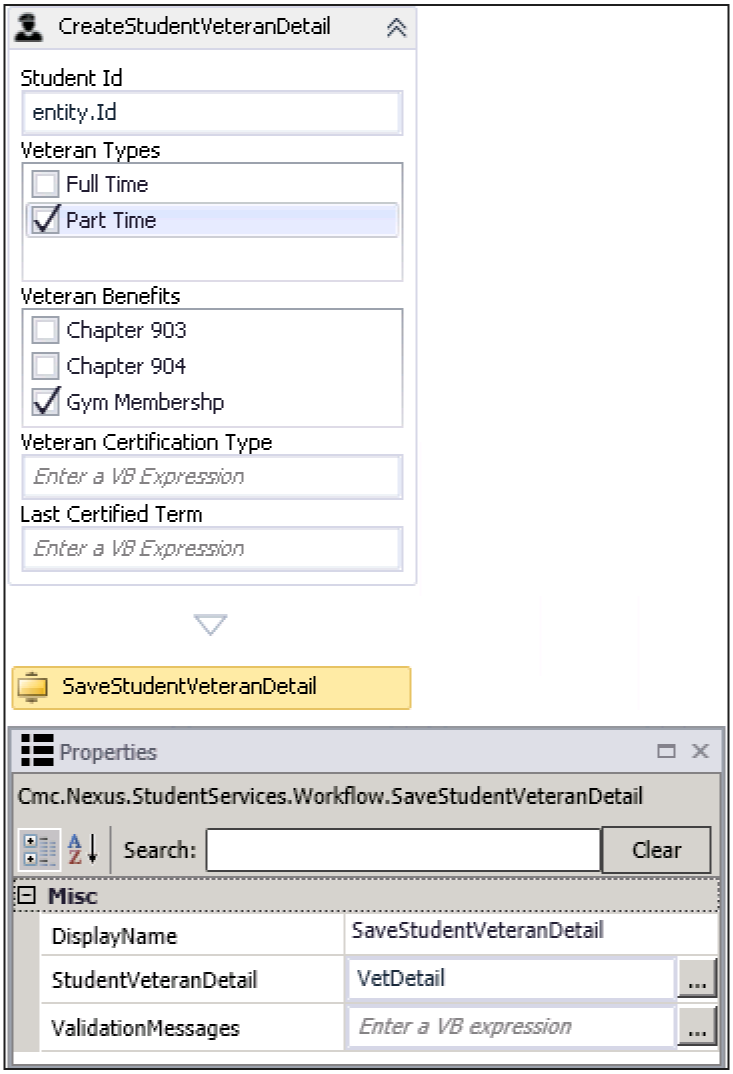
SaveStudentSportsService Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
StudentAthleticDetail	`InOutArgument <StudentAthleticDetailEntity>`	Yes	Specify the StudentAthleticDetail entity to be saved using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentServices.Entities**, select **StudentAthleticDetailEntity**, and click **OK**.  Name Variable type Sport Cmc.Nexus.StudentServices.Entities.StudentAthleticDetailEntity See StudentAthleticDetailEntity Class in the Anthology Student Object Library.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

SaveStudentVeteranDetail (V2)

The SaveStudentVeteranDetail activity saves a Student Veteran Detail record that was created with the CreateStudentVeteranDetail (V2) activity.


CreateStudentVeteranDetail

Student Id

entity.Id

Veteran Types

□ Full Time
☑ Part Time

Veteran Benefits

□ Chapter 903
□ Chapter 904
☑ Gym Membershp

Veteran Certification Type

Enter a VB Expression

Last Certified Term

Enter a VB Expression

SaveStudentVeteranDetail

Properties

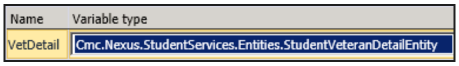
Cmc.Nexus.StudentServices.Workflow.SaveStudentVeteranDetail

DisplayName	SaveStudentVeteranDetail
StudentVeteranDetail	VetDetail
ValidationMessages	*Enter a VB expression*

Properties

SaveStudentVeteranDetail Properties

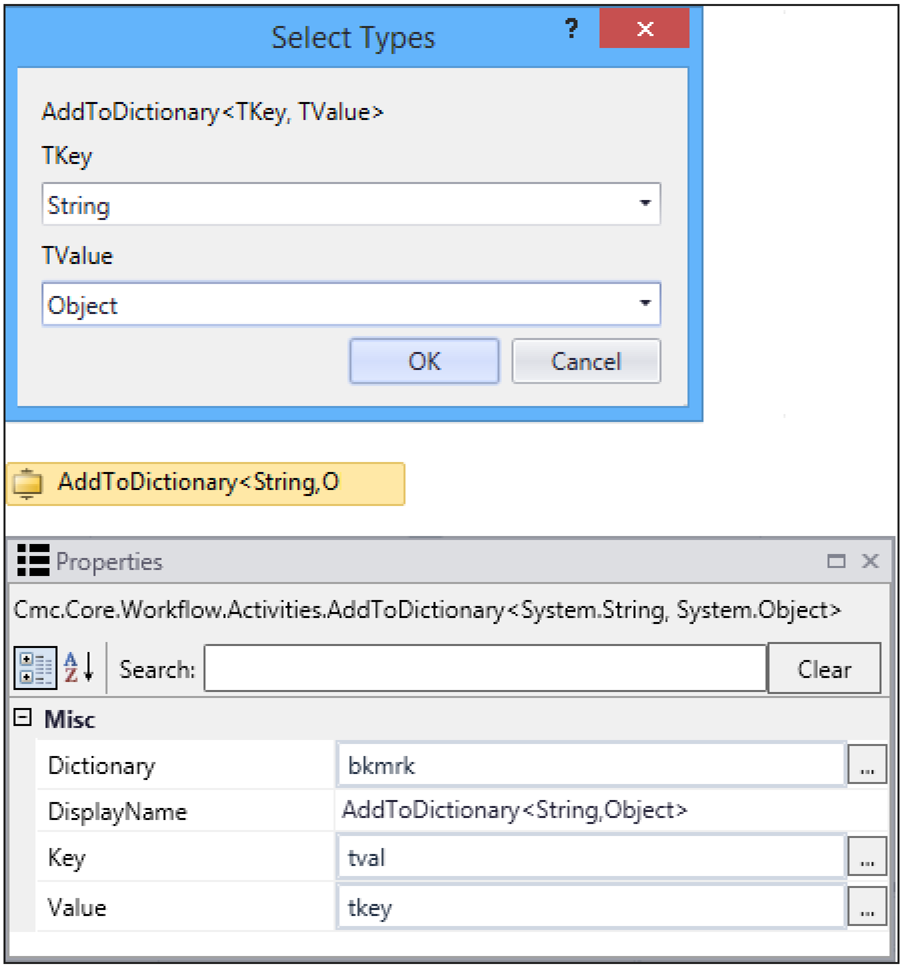
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
StudentVeteranDetail	`InOutArgument <StudentVeteranDetailEntity>`	Yes	Specify the StudentVeteranDetail entity to be saved using a VB expression or variable. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to **Cmc.Nexus.StudentServices.Contracts > Cmc.Nexus.StudentServices.Entities**, select **StudentVeteranDetailEntity**, and click OK.  Name Variable type VetDetail Cmc.Nexus.StudentServices.Entities.StudentVeteranDetailEntity See StudentVeteranDetailEntity Class in the Anthology Student Object Library.
ValidationMessages	`InOutArgument <ValidationMessageCollection>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

Cmc.Core.Workflow.Activities

AddToDictionary<>

The AddToDictionary<> activity maps a key type (TKey) to a value type (TValue) in the dictionary. You select the .NET data type for the TKey and TValue, for example, Int32, String, Boolean, Array, Object, etc.



AddToDictionary<TKey, TValue>

TKey: String

TValue: Object

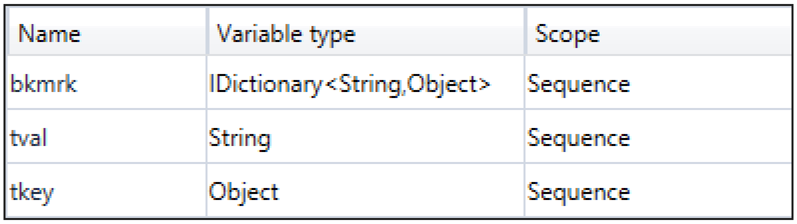
AddToDictionary<String,O

Cmc.Core.Workflow.Activities.AddToDictionary<System.String, System.Object>

Misc

Dictionary	bkmrk
DisplayName	AddToDictionary<String,Object>
Key	tval
Value	tkey

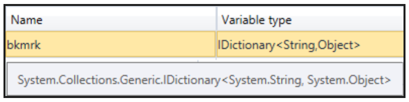
This workflow example uses the following variable definitions:



Name	Variable type	Scope
bkmrk	IDictionary<String,Object>	Sequence
tval	String	Sequence
tkey	Object	Sequence

Properties

AddToDictionary<> Properties

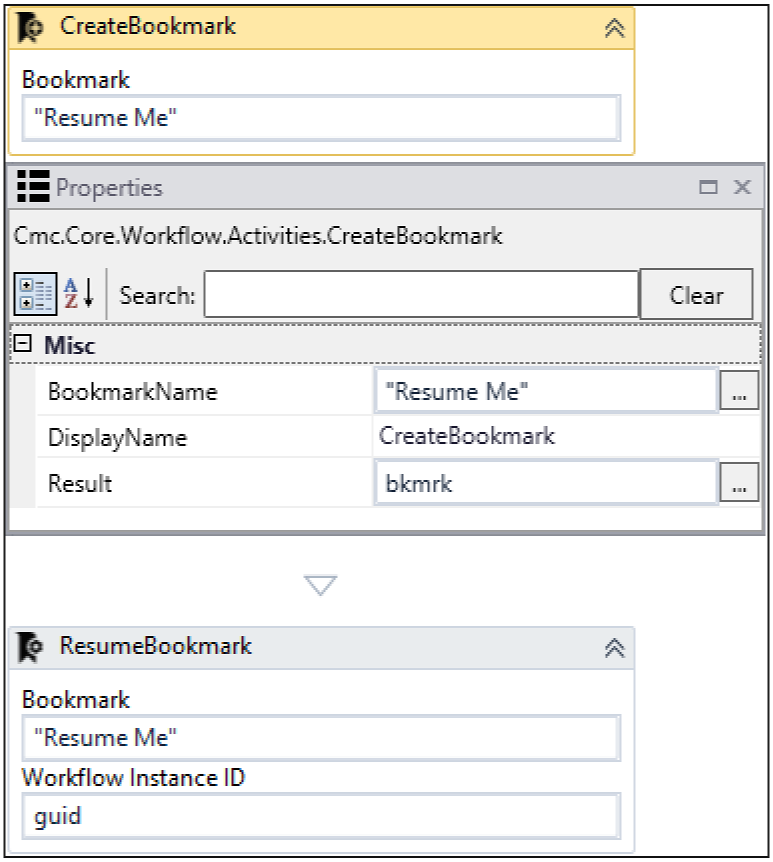
Property	Value	Required	Notes
Dictionary	`InArgument<IDictionary<selected data type,selected data type>>`	Yes	Specify the Dictionary using a VB expression or variable. Refer to the image below for the Variable type selection.  Name | Variable type bkmrk | IDictionary<String,Object> System.Collections.Generic.IDictionary<System.String, System.Object>
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Key	`InArgument<selected data type>`	Yes	Specify the Key using a VB expression or variable. Select the data type when you add the activity to the workflow.
Value	`InArgument<selected data type>`	Yes	Specify the Value using a VB expression or variable. Select the data type when you add the activity to the workflow.

To see how AddToDictionary<> can be used in a workflow, refer to:

- Populate Fields in a Forms Builder Form

CreateBookmark

The CreateBookmark activity creates a named bookmark in a workflow at the point where the workflow execution can be resumed at a later time. This activity is used to persist a workflow instance. Once a workflow is persisted, it can continue execution using the ResumeBookmark activity or the IWorkflowEngine::ResumeBookmark method in .NET.



CreateBookmark

Bookmark: "Resume Me"

Cmc.Core.Workflow.Activities.CreateBookmark

Misc

BookmarkName	"Resume Me"
DisplayName	CreateBookmark
Result	bkmrk

ResumeBookmark

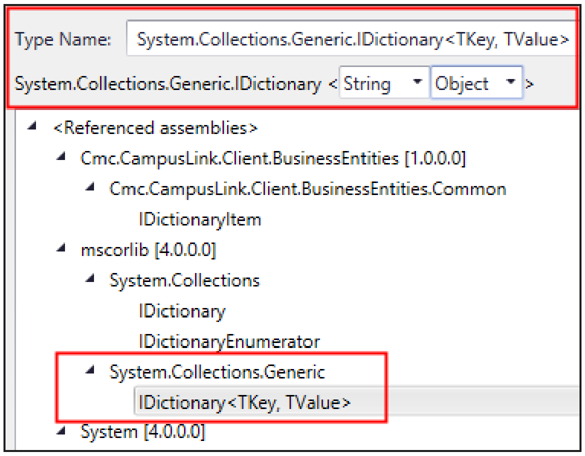
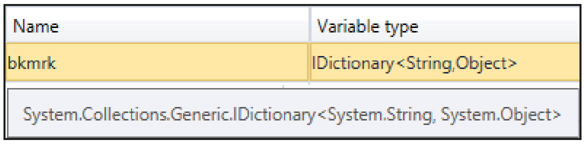
Bookmark: "Resume Me"

Workflow Instance ID: guid

Properties

CreateBookmark Properties

Property	Value	Required	Notes
BookmarkName	`InArgument<String>`	Yes	Specify the BookmarkName using a VB expression or variable. More than one bookmark can be executing at a time; therefore, this property is used to uniquely identify the bookmark associated with this activity.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
Result	`OutArgument <IDictionary <String,Object>>`	Yes	Specify the Result using a VB expression or variable. The Result value is passed from a call to ResumeBookmark or IWorkflowEngine::ResumeBookmark. The following image shows how to browse and select the variable type.  Type Name: System.Collections.Generic.IDictionary<TKey, TValue> System.Collections.Generic.IDictionary < String Object > <Referenced assemblies> Cmc.CampusLink.Client.BusinessEntities [1.0.0.0] Cmc.CampusLink.Client.BusinessEntities.Common IDictionaryItem mscorlib [4.0.0.0] System.Collections IDictionary IDictionaryEnumerator System.Collections.Generic IDictionary<TKey, TValue> System [4.0.0.0]  Name: bkmrk Variable type: IDictionary<String,Object> System.Collections.Generic.IDictionary<System.String, System.Object>

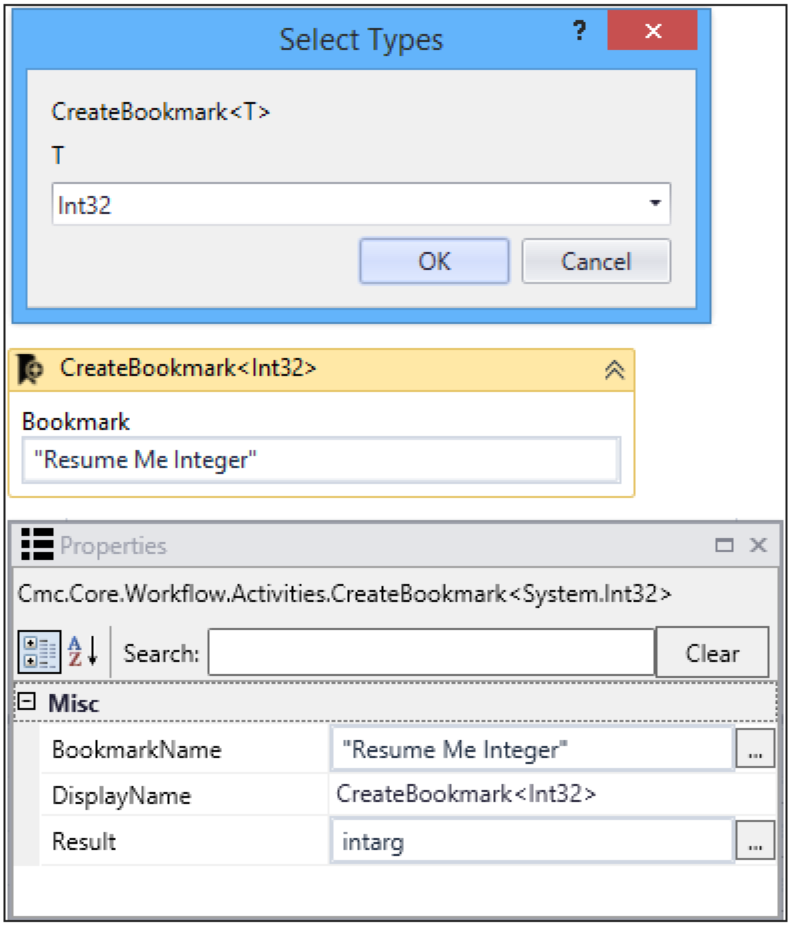
To see how CreateBookmark can be used in a workflow, refer to:

- Create a Long Running Workflow

CreateBookmark<>

The CreateBookmark<> activity creates a named bookmark where the workflow execution can be resumed at a later time and through which data can be delivered.

The only difference between CreateBookmark and CreateBookmark<> is that CreateBookmark<> allows an input argument. You select the .NET data type for the input, for example, Int32, String, Boolean, Array, Object, etc.



Properties

CreateBookmark<> Properties

Property	Value	Required	Notes
BookmarkName	`InArgument<String>`	Yes	Specify the BookmarkName using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

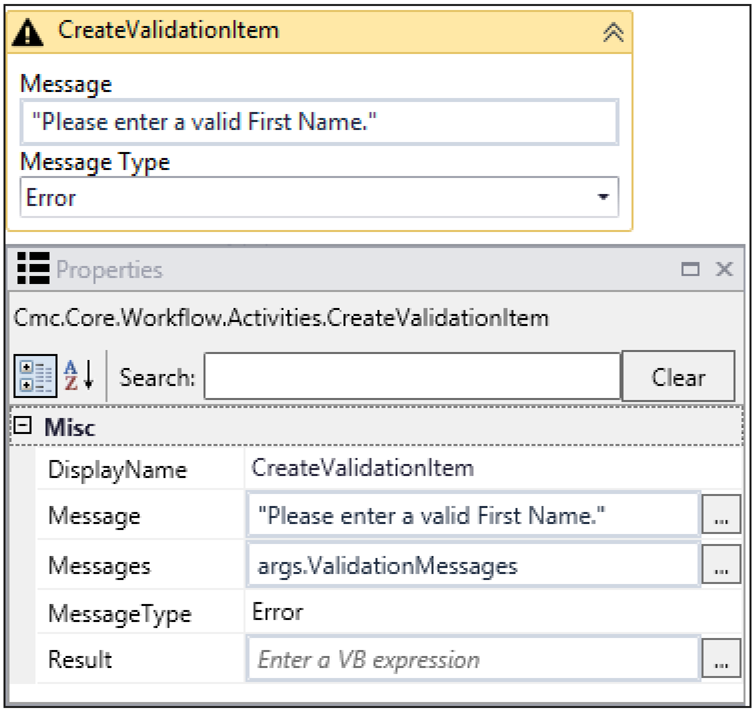
Property	Value	Required	Notes
Result	`OutArgument<selected data type>`	Yes	Specify the Result using a VB expression or variable. Select the data type when you add the activity to the workflow.

CreateValidationItem

The CreateValidationItem activity enables you to display a message in the UI when a workflow is executed.

⚠ This activity can only be used with Saving events.

If the same event triggers multiple validation items, the validation messages are consolidated in one message box titled "Custom Validation Message".



CreateValidationItem

Message: "Please enter a valid First Name."

Message Type: Error

Properties

Cmc.Core.Workflow.Activities.CreateValidationItem

Search:

Misc

DisplayName	CreateValidationItem
Message	"Please enter a valid First Name."
Messages	args.ValidationMessages
MessageType	Error
Result	*Enter a VB expression*

Properties

CreateValidationItem Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Message	`InArgument<String>`	Yes	Specify the text of the validation message, for example: `"Please enter a mobile phone number."`

Property	Value	Required	Notes
Messages	`InArgument <ICollection<ValidationMessage>>`	Yes	In the Messages field of the Properties pane, enter the following VB code: `args.ValidationMessages`
Message Type	`ValidationMessageType`	Yes	Select a value in the drop-down list of the activity in the Designer window. The options are: • Error • Information • Warning
Result	`OutArgument<ValidationMessage>`	No	If necessary, specify the out argument using a VB expression or variable.

To see how CreateValidationItem can be used in a workflow, refer to:

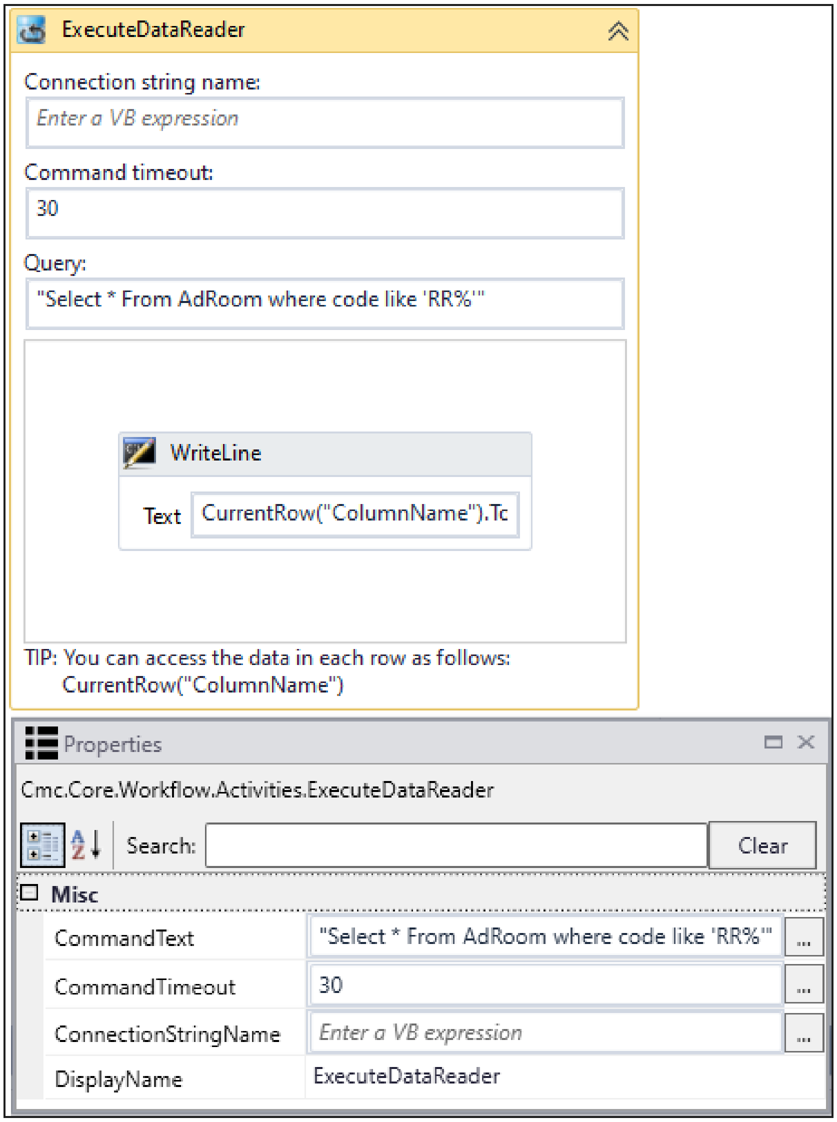
- Custom Field Validations on Each Step of Enrollment Wizard.

ExecuteDataReader

The ExecuteDataReader activity enables you to create workflows that perform two steps:

1. Execute an SQL query.
2. Execute activities in the query result.

If the query successfully connects to the data source, it queries the database and executes the activities in the body once per data row returned.



ExecuteDataReader

Connection string name: *Enter a VB expression*

Command timeout: 30

Query: "Select * From AdRoom where code like 'RR%'"

WriteLine

Text CurrentRow("ColumnName").Tc

TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

Properties

Cmc.Core.Workflow.Activities.ExecuteDataReader

Search: Clear

Misc

CommandText	"Select * From AdRoom where code like 'RR%'"
CommandTimeout	30
ConnectionStringName	*Enter a VB expression*
DisplayName	ExecuteDataReader

In general, the connection strings used during workflow execution are retrieved from the web.config of the product that triggers workflow execution.

Only if you want to run a workflow with ExecuteDataReader, ExecuteNonQuery, or ExecuteQuery activity in test mode using the **Run** option in Workflow Composer, would you need to manually add the connection string to the Workflow Composer web.config file.

Properties

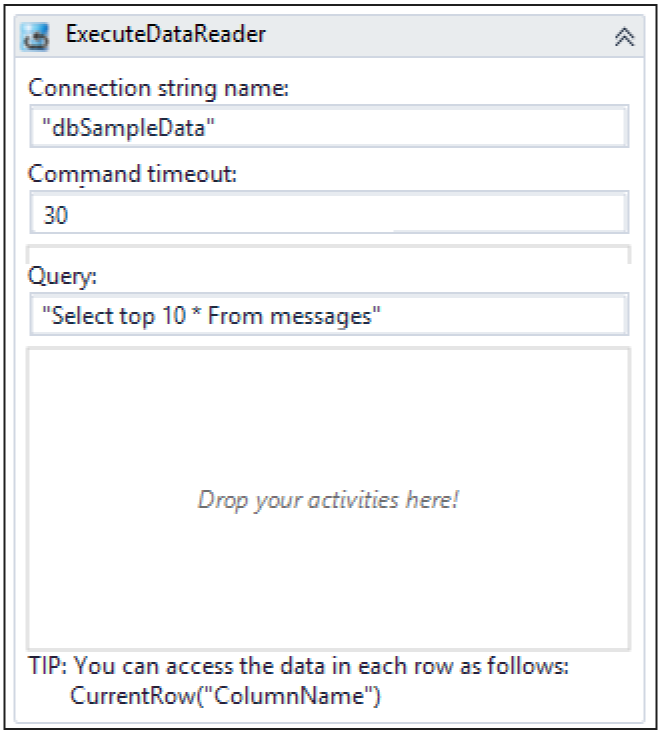
ExecuteDataReader Properties

Property	Value	Required	Notes
CommandText	`InArgument<String>`	Yes	Enter a command that specifies the query to perform on the target data source and is expected to return a result set. **Note**: Supply an SQL query that will only return one set of rows from one table. Do not attempt to return multiple sets of data since this activity will only utilize the first set of data rows returned. *Example* `"Select * from Messages"`
CommandTimeout	`InArgument<Int32>`	No	You can adjust the CommandTimeout value if the activity needs to execute long-running SQL statements. The default and minimum command timeout is 30 seconds. The maximum is 1800 (30 minutes).
ConnectionStringName	`InArgument<String>`	No	Enter the name of a connection string that has been configured in the CONFIG file of the host application that is executing the workflows (see Connection Strings). If none is specified, this activity attempts to connect to a connection string named `DbConnection`. **Note**: Forms Builder 3.6 introduces the "CrmConnection" string in the web.config of Forms Renderer (see Renderer Connection Strings). If you have created workflows with ExecuteDataReader activities, ensure that connection strings in the activities match the updated web.config of Forms Renderer.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

ExecuteDataReader Example 1

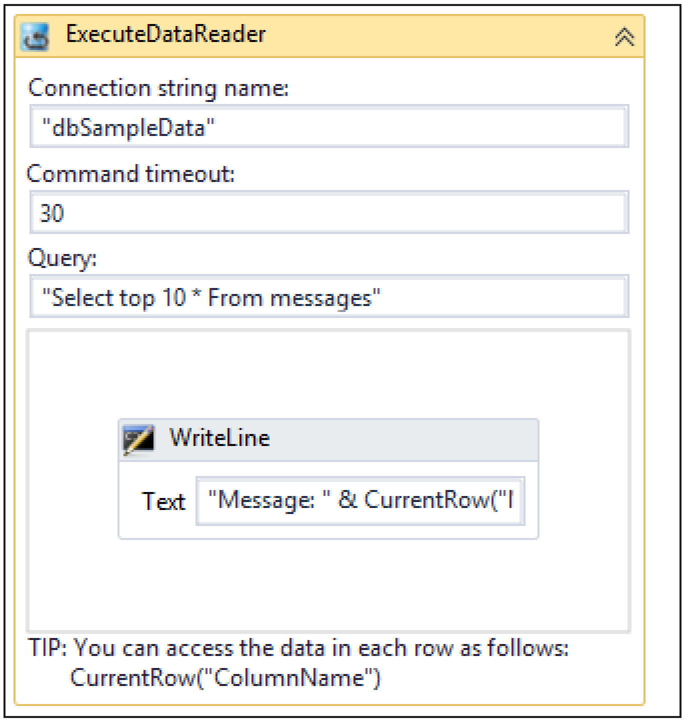
This example retrieves rows from the database and writes the results to the console.

1. Open a workflow or create a new workflow.
2. Drag the **ExecuteDataReader** activity into your workflow.
3. Specify the values for the input arguments or map them to workflow variables.



ExecuteDataReader
Connection string name:
"dbSampleData"
Command timeout:
30
Query:
"Select top 10 * From messages"
Drop your activities here!
TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

4. Add activities into the body of this activity.



Tip: The activities in the body of this activity will be executed once per every row returned from the database query.

You can access the data in each row as a variable called `CurrentRow`.

You can then use the data in each row using the format: `CurrentRow("ColumnName")`.

5. Run the workflow.

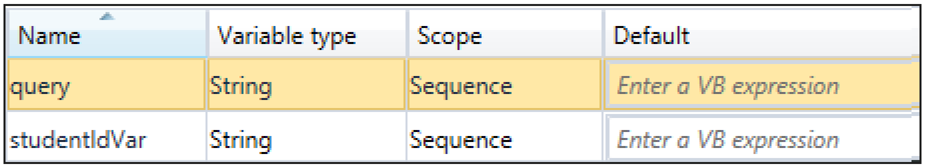
   Result:

   The query successfully connects to the data source, queries the database, and executes the activities in the body once per data row returned.

ExecuteDataReader Example 2

This example retrieves a value from a single row in the database and uses the retrieved value in an assignment statement.

1. Open a workflow or create a new workflow.
2. Create two variables to hold the query statement and the value retrieved from the database.
   - **query**
   - **studentIdVar**

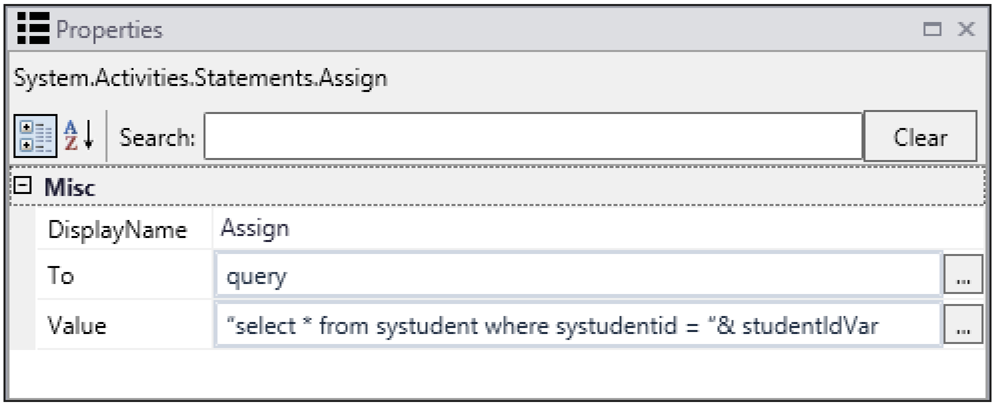


Name	Variable type	Scope	Default
query	String	Sequence	Enter a VB expression
studentIdVar	String	Sequence	Enter a VB expression

3. Drag an **Assign** activity into a sequence.

   Assign the following value to a string named **query**:

   "select * from systudent where systudentid = "& studentIdVar



System.Activities.Statements.Assign

Misc

DisplayName	Assign
To	query
Value	"select * from systudent where systudentid = "& studentIdVar

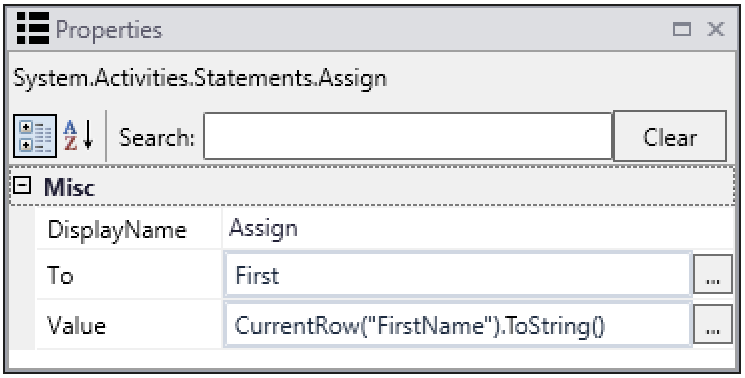
4. Drag the **ExecuteDataReader** activity into your sequence.

5. In the Query field of the ExecuteDataReader activity, specify **query** (the name of the string assigned in the previous step).

6. Drop an **Assign** activity into the body of the ExecuteDataReader activity.

   Assign the following value to a string named **First**:

   CurrentRow("FirstName").toString()

   Properties

   System.Activities.Statements.Assign

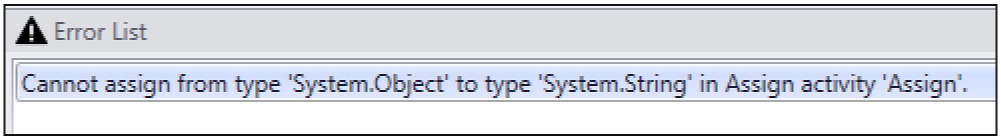
DisplayName	Assign
To	First
Value	CurrentRow("FirstName").ToString()

   Note:

   The data type returned by the query must be specified in the assignment.

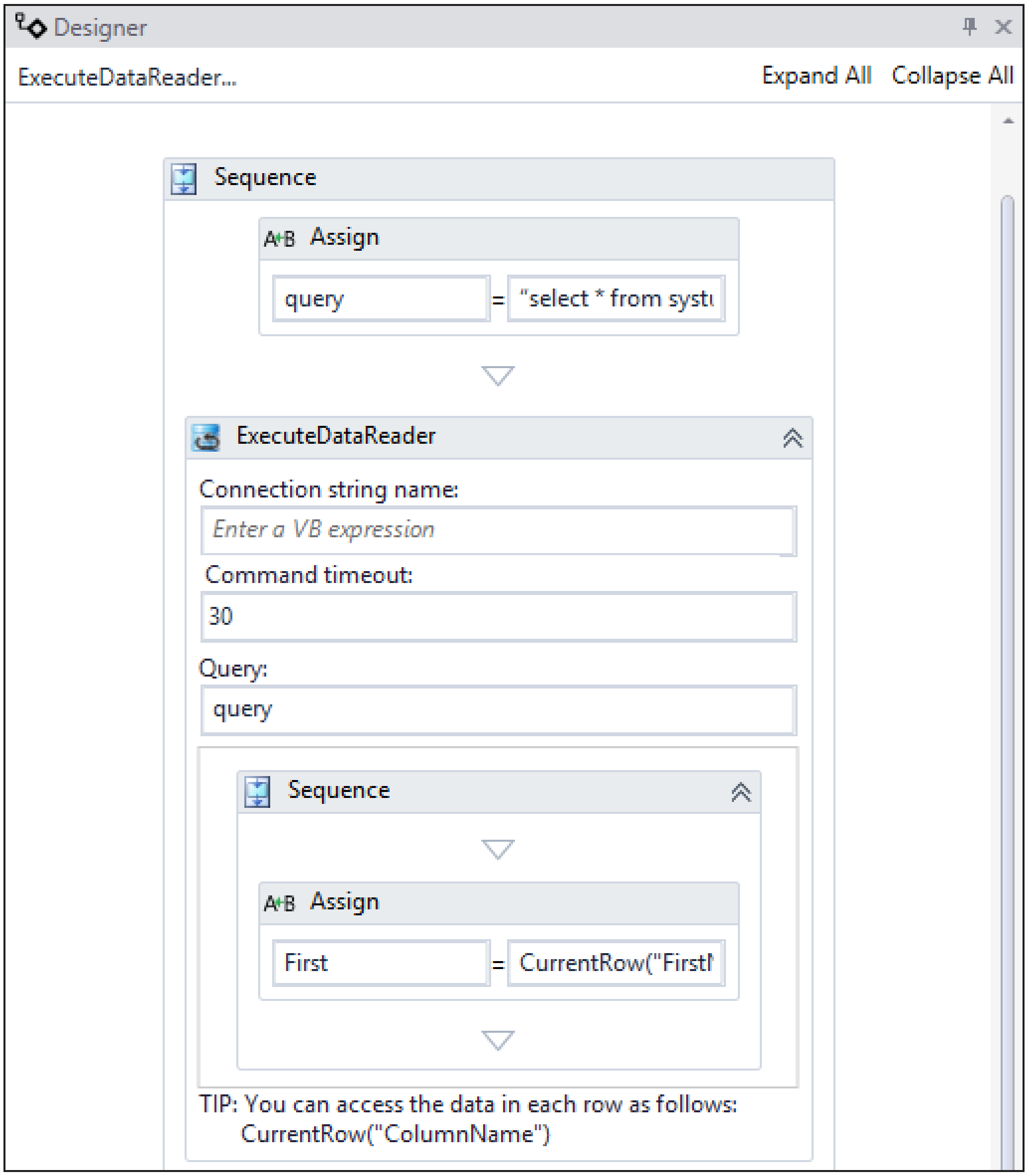
   - To get a string field value from a database row, the expression**.ToString()** is needed.
   - To get an integer value, the assignment would be like this: **Convert.ToInt32(CurrentRow(“dbIntegerField”))**

   Without the type conversion, the assignment statement fails with the following error:

   Cannot assign from type 'System.Object' to type 'System.String' in Assign activity 'Assign'.

   The following image shows the completed workflow section:



query = "select * from systı

Connection string name: *Enter a VB expression*

Command timeout: 30

Query: query

First = CurrentRow("FirstI

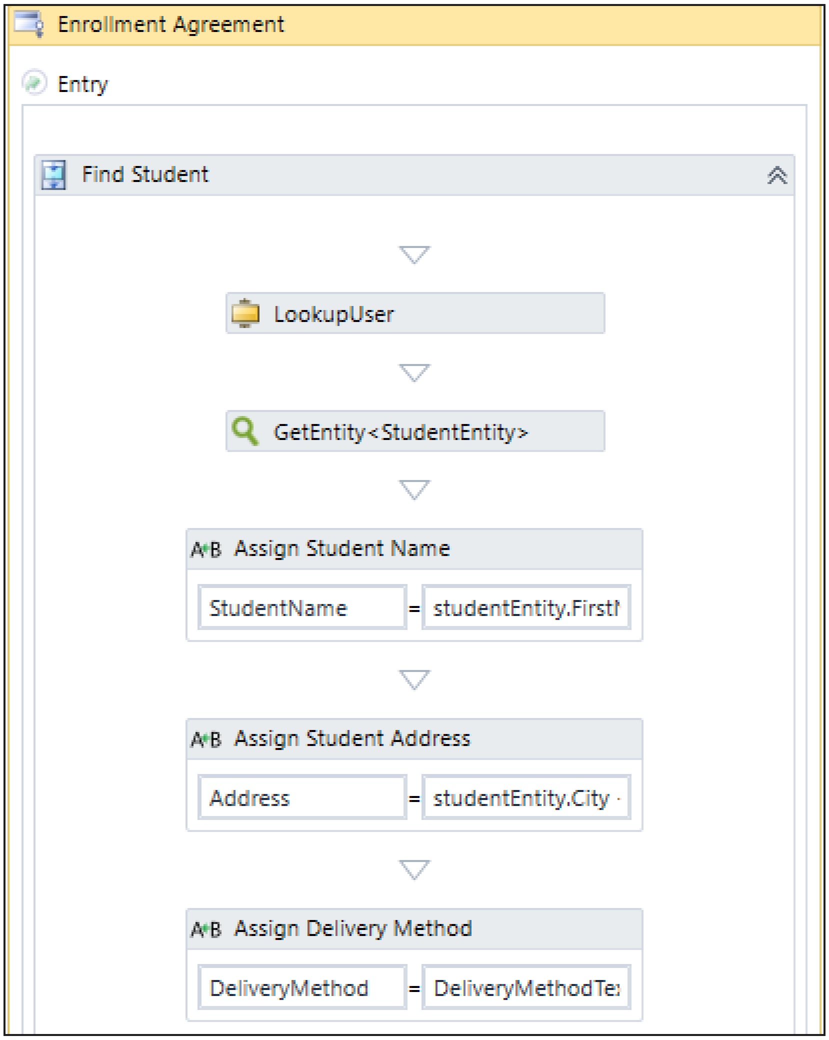
TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

ExecuteDataReader Example 3

The following example uses the ExecuteDataReader activity in the context of a Forms Builder sequence. The form sequence prompts the user to enter his/her contact details, select a program, and e-sign an enrollment agreement.

1. The LookupUser activity captures the UserName from the formInstance.UserName argument and returns the studentid.
2. The GetEntity activity takes the studentid value and returns the studentEntity value.
3. Two Assign activities associate the student name and address fields of the studentEntity with values passed from the form sequence via the StudentName and Address arguments.

4. The third Assign activity associates the DeliveryMethod argument from the form sequence with the DeliveryMethodText variable in the workflow. The variable can be assigned a default string, e.g., "Program is blended (hybrid) including on-ground and online delivery".



Enrollment Agreement

Entry

Find Student

LookupUser

GetEntity<StudentEntity>

Assign Student Name

StudentName = studentEntity.FirstN

Assign Student Address

Address = studentEntity.City

Assign Delivery Method

DeliveryMethod = DeliveryMethodTe:

5. The ExecuteDataReader runs the following query on the Anthology Student database to retrieve the enrollment ID for the student ID:

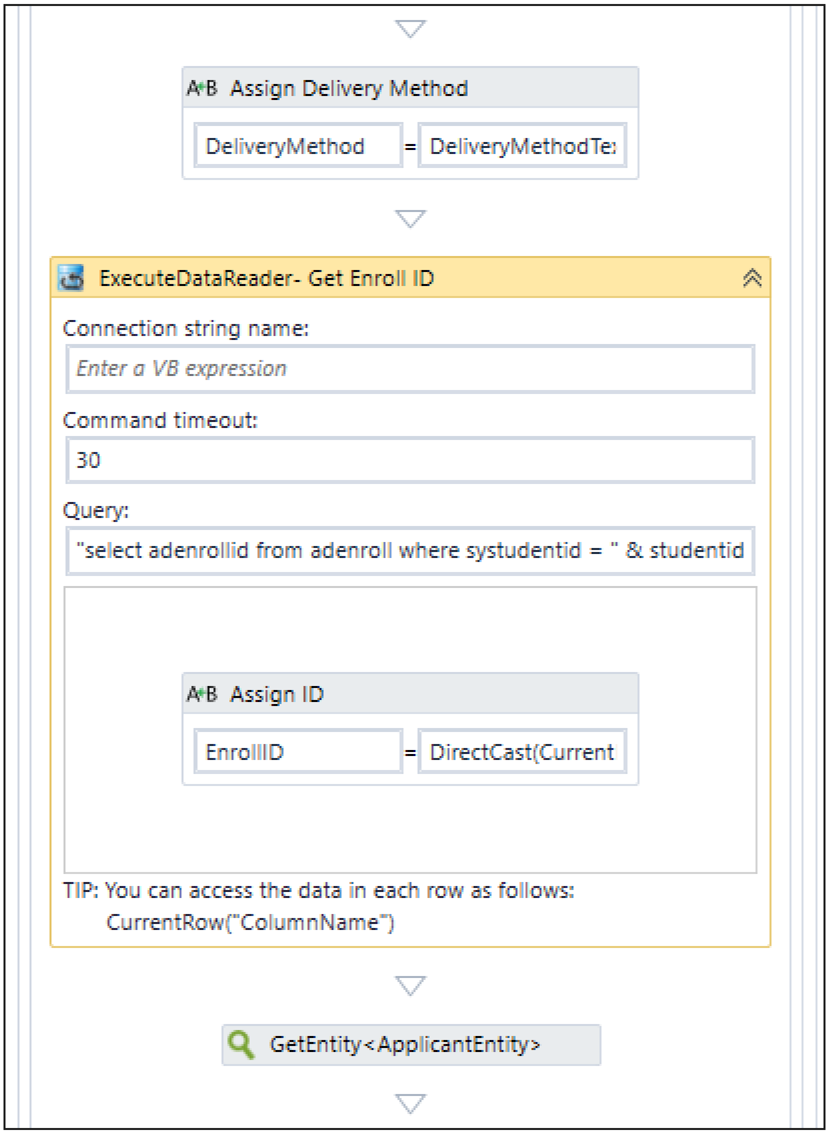
   "select adenrollid from adenroll where systudentid = " & studentid

   In Anthology Student, the adenrollid from the AdEnroll table is used as the enrollment identifier if an applicant record is converted to an enrollment record.

6. The Assign activity within in the ExecuteDataReader assigns the following value to the EnrollID:

   DirectCast(CurrentRow("adenrollid"), int32)

7. The GetEntity activity below the ExecuteDataReader uses the EnrollID value to retrieve the ApplicantEntity.



A+B Assign Delivery Method

DeliveryMethod = DeliveryMethodTe:

ExecuteDataReader- Get Enroll ID

Connection string name:

Enter a VB expression

Command timeout:

30

Query:

"select adenrollid from adenroll where systudentid = " & studentid

A+B Assign ID

EnrollID = DirectCast(Current

TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

GetEntity<ApplicantEntity>

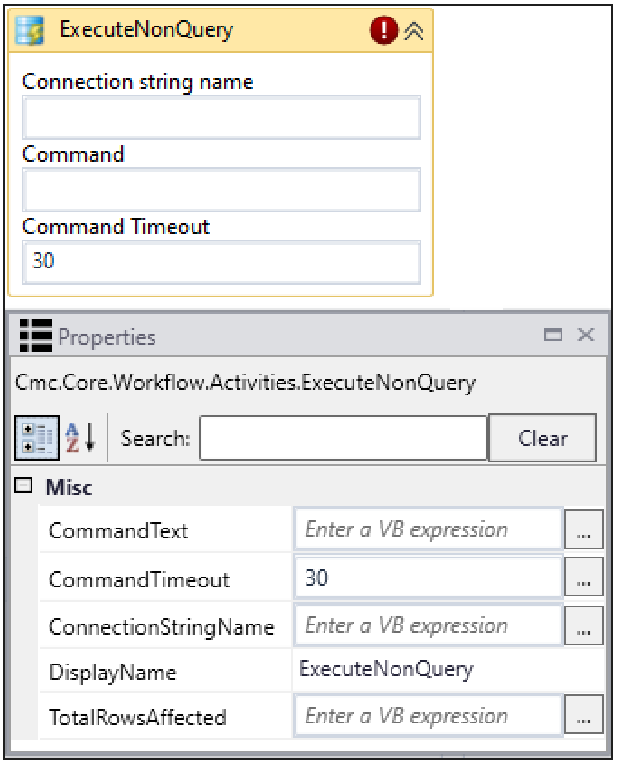
8. Next, the workflow retrieves the enrollment agreement document for the student from the database, presents the document to the student for e-signature, and saves the signed document.

For more examples of workflows with ExecuteDataReader, see:

- Register Students into a Course
- Wake up the Long Running Workflow

ExecuteNonQuery

The ExecuteNonQuery activity enables you to execute SQL statements that INSERT, UPDATE, or DELETE data in a given data source. For more information, see ExecuteNonQuery Example.



Connection string name

Command

Command Timeout
30

Cmc.Core.Workflow.Activities.ExecuteNonQuery

CommandText	*Enter a VB expression*
CommandTimeout	30
ConnectionStringName	*Enter a VB expression*
DisplayName	ExecuteNonQuery
TotalRowsAffected	*Enter a VB expression*

In general, the connection strings used during workflow execution are retrieved from the web.config of the product that triggers workflow execution.

Only if you want to run a workflow with ExecuteDataReader, ExecuteNonQuery, or ExecuteQuery activity in test mode using the **Run** option in Workflow Composer, would you need to manually add the connection string to the Workflow Composer web.config file.

Properties

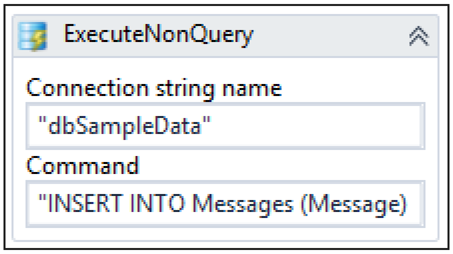
ExecuteNonQuery Properties

Property	Value	Required	Notes
CommandText	`InArgument<String>`	Yes	Enter a command that specifies the activity to perform on the target data source - and should not be expected to return a result set. This activity allows commands that INSERT, UPDATE, or DELETE records in the target database. *Example* `INSERT INTO Messages (Message) VALUES ('New message added')`
CommandTimeout	`InArgument<Int32>`	No	You can adjust the CommandTimeout value if the activity needs to execute long-running SQL statements. The default and minimum command timeout is 30 seconds. The maximum is 1800 (30 minutes).
ConnectionStringName	`InArgument<String>`	Yes	Enter the name of a connection string that has been configured in the CONFIG file of the host application that is executing the workflows (see Connection Strings). If none is specified, this activity attempts to connect to a connection string named `DbConnection`. **Note**: Forms Builder 3.6 introduces the "CrmConnection" string in the web.config of Forms Renderer (see Renderer Connection Strings). If you have created workflows with ExecuteNonQuery activities, ensure that connection strings in the activities match the updated web.config of Forms Renderer.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
TotalRowsAffected	`OutArgument<Int32>`	Yes	The output argument contains the total number of rows affected by the execution of the SQL command in the database. *Example* If a DELETE command was entered as input argument and 12 rows were deleted from a table, the resulting value is '12'.

ExecuteNonQuery Example

1. Open a workflow or create a new workflow.
2. Drag the **ExecuteNonQuery** activity into your workflow.
3. Specify the values for the input arguments or map them to workflow variables.



ExecuteNonQuery

Connection string name

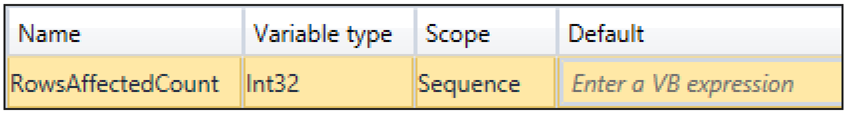
"dbSampleData"

Command

"INSERT INTO Messages (Message)

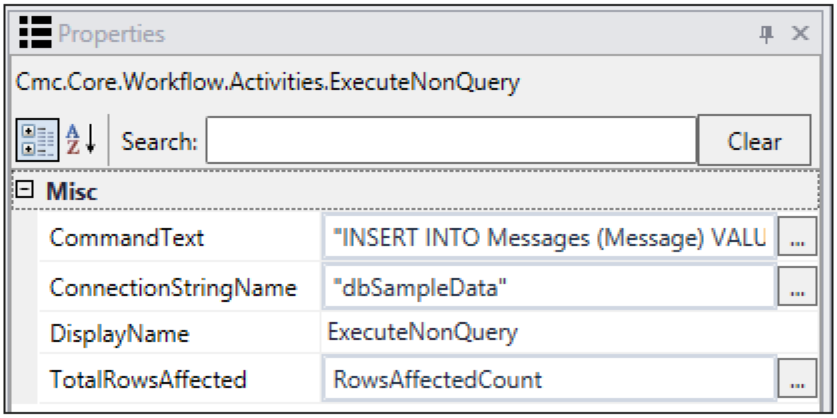
4. Create a workflow variable of data type **Int32** that will be mapped to the result of the query execution.

   In this example, we created a new variable called RowsAffectedCount.



Name	Variable type	Scope	Default
RowsAffectedCount	Int32	Sequence	*Enter a VB expression*

5. Configure the output argument in the activity named TotalRowsAffected to the new workflow variable RowsAffectedCount.



Cmc.Core.Workflow.Activities.ExecuteNonQuery

Misc

CommandText	"INSERT INTO Messages (Message) VALU
ConnectionStringName	"dbSampleData"
DisplayName	ExecuteNonQuery
TotalRowsAffected	RowsAffectedCount

6. Run the workflow.

   Result: If the query successfully connects to the data source, it populates your local variable with the total rows affected by the query.

To see how ExecuteNonQuery can be used in a workflow, refer to:

- Create a Long Running Workflow

ExecuteODataQuery<>

⚠ **Prerequisite**: When this activity is used with Anthology Student 21.2.0 and later, the APIUser must have authorization to access to the entity requested in the OData query. For more information, see Security Enhancement for OData Queries.

The ExecuteODataQuery<> activity returns an OData query against the query model. The result of the OData query can be used as input for subsequent workflow activities.

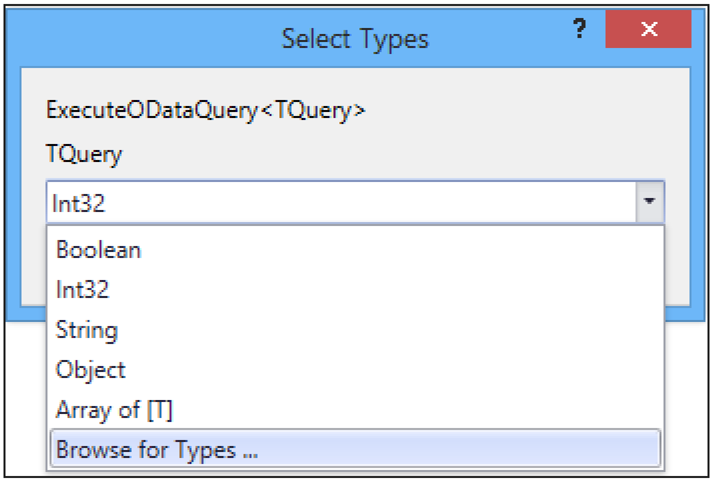
The results of the OData query are available in the body of the activity. The results are iterated, and each item returned is available using the **item** variable. The ExecuteODataQuery<> is useful when you want to retrieve a list of students, courses, or any entity, and you want to perform the same activity using each entity returned in the list.

For example, you could use this activity to:

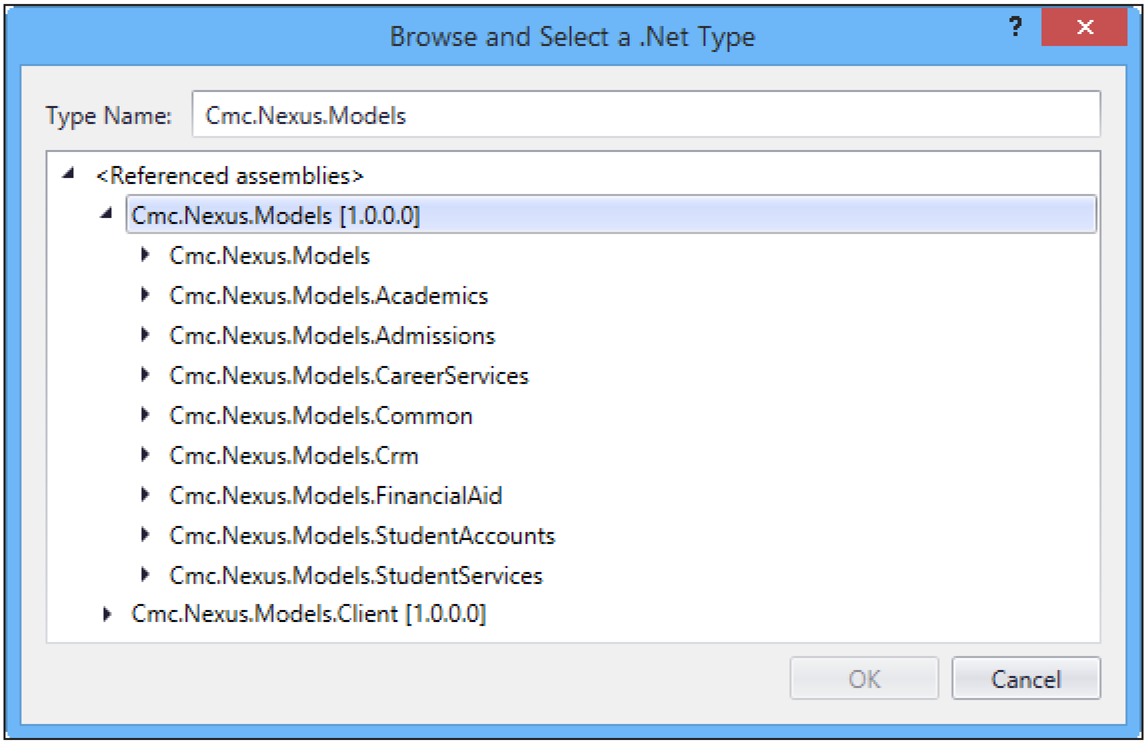
- Get a class roster for student’s in a class section and send an email to each student.
- Get a list of all student’s in a student group or hold group and charge each one a late fee.

Note: This activity does not return entities. A conversion assignment needs to be made to bind query result to the entity model. Once bound, the data can be edited. For an example of how to bind OData query results to a grid in a form sequence, see the Forms Builder help topic Grid Bound to Results of ExecuteODataQuery Activity.

When ExecuteODataQuery<> activity is dropped into the Designer pane, the workflow dialog prompts for the **<TQuery>**, which is the model type to query against. The **Browse for Type...** option must be used to select the query model type.



All query model types are located in the **Cmc.Nexus.Models** namespace.



Type Name: Cmc.Nexus.Models

- <Referenced assemblies>
  - Cmc.Nexus.Models [1.0.0.0]
    - Cmc.Nexus.Models
    - Cmc.Nexus.Models.Academics
    - Cmc.Nexus.Models.Admissions
    - Cmc.Nexus.Models.CareerServices
    - Cmc.Nexus.Models.Common
    - Cmc.Nexus.Models.Crm
    - Cmc.Nexus.Models.FinancialAid
    - Cmc.Nexus.Models.StudentAccounts
    - Cmc.Nexus.Models.StudentServices
  - Cmc.Nexus.Models.Client [1.0.0.0]

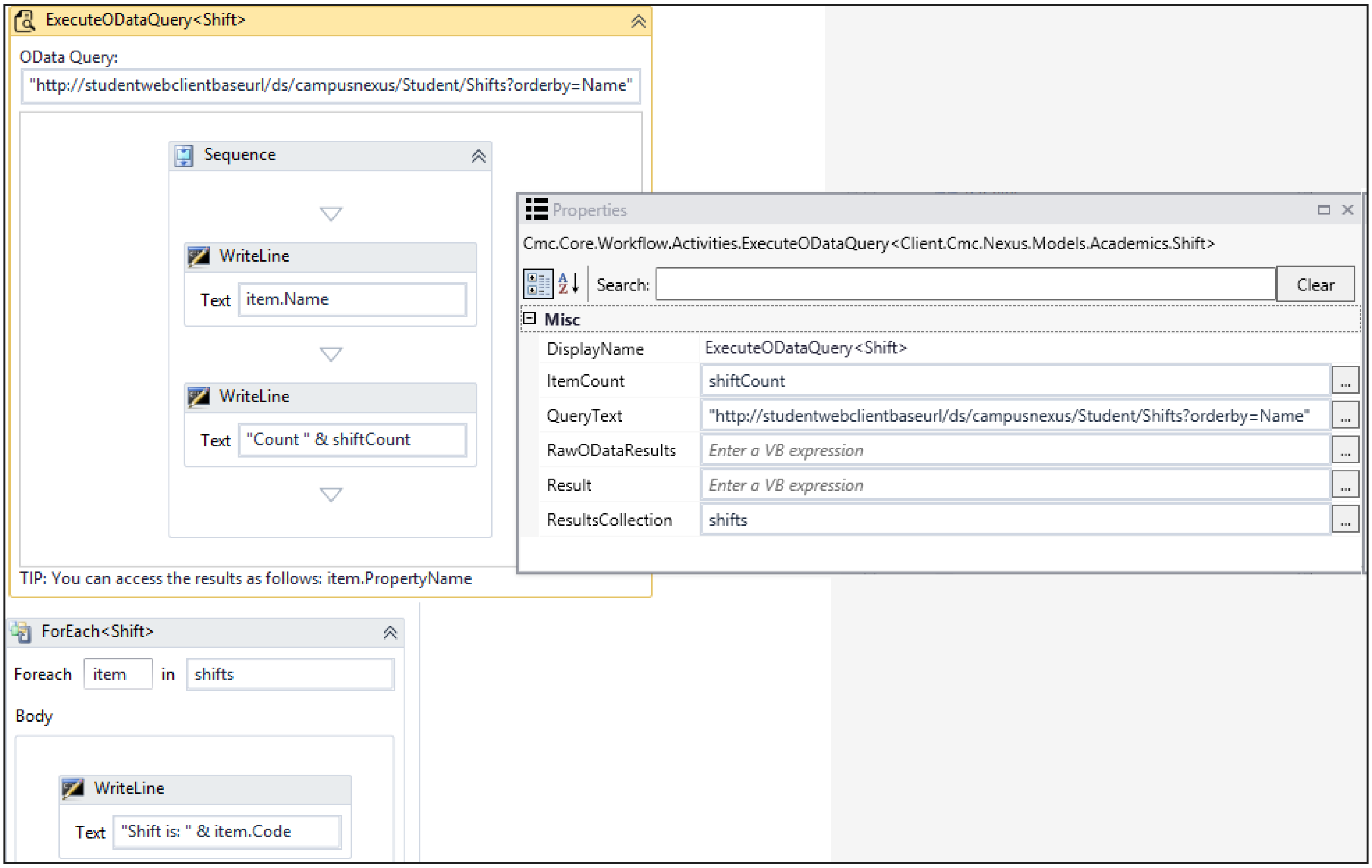
The model type selected in the Cmc.Nexus.Models namespace must match the primary type for your OData query. For example, if the primary type in the OData query is <Students>, the TQuery model type must also be <Students>.

Note: The Cmc.Nexus.Models.{module}.{type} namespace for the selected model type must be available in Workflow Composer so that the results of the OData query can be assigned. Otherwise an error message will be displayed. To avoid errors such as *"'<entity>' is not defined"*, in the body of the ExecuteODataQuery<> activity, add any activity (e.g., WriteLine or LogLine) with specific value to be written or logged. e.g., item.FirstName (or whatever is applicable for the OData query). Adding an activity ensures that the Imports required for the selected namespace are set properly. If you do not include an activity in the body of the ExecuteODataQuery<> activity, add the required namespace to the Imports tab in Workflow Composer.

To create an OData query, you can use the Web Client for Anthology Student. It is easiest to get the OData query results in a browser first to verify that the query is valid and that it returns the expected results. Once you have the desired query and results, paste the query into the **QueryText** field in the Properties pane of the ExecuteODataQuery<> activity. Enclose the query string in double quotes.

Example

"http://<localhost>/Cmc.Nexus.Web/ds/campusnexus/Student/Shifts?$orderby=Name"



ExecuteODataQuery<Shift>

OData Query: "http://studentwebclientbaseurl/ds/campusnexus/Student/Shifts?orderby=Name"

Sequence

WriteLine
Text: item.Name

WriteLine
Text: "Count " & shiftCount

TIP: You can access the results as follows: item.PropertyName

Properties

Cmc.Core.Workflow.Activities.ExecuteODataQuery<Client.Cmc.Nexus.Models.Academics.Shift>

Search: | Clear

Misc

DisplayName	ExecuteODataQuery<Shift>
ItemCount	shiftCount
QueryText	"http://studentwebclientbaseurl/ds/campusnexus/Student/Shifts?orderby=Name"
RawODataResults	*Enter a VB expression*
Result	*Enter a VB expression*
ResultsCollection	shifts

ForEach<Shift>

Foreach item in shifts

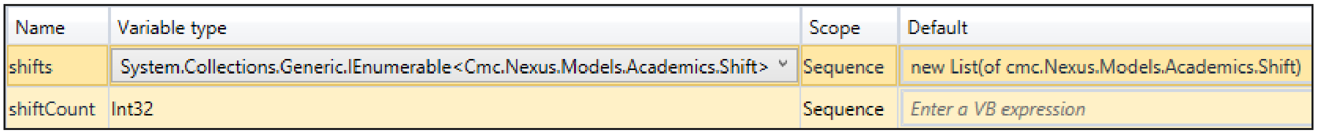
Body

WriteLine
Text: "Shift is: " & item.Code

Optionally, the query results can also be stored in variables.

- The raw JSON string result is available via the **RawODataResults** property.
- If the total count of items returned by the query is needed, the **ItemCount** property can be used.
- The collection of items can be saved to a variable by providing the properly typed variable in the **ResultsCollection** property.

  For the ExecuteODataQuery<Shift> example, the variable type needs to be **IEnumerable<Shift>**. The variable must have a default value of **new List(of cmc.Nexus.Models.Academics.Shift)**.

Name	Variable type	Scope	Default
shifts	System.Collections.Generic.IEnumerable<Cmc.Nexus.Models.Academics.Shift>	Sequence	new List(of cmc.Nexus.Models.Academics.Shift)
shiftCount	Int32	Sequence	*Enter a VB expression*

  To view the query result with the above property settings in the Output pane, insert a WriteLine activity that displays **"Count " & shiftCount**.

Navigational properties are also available in the results. For example, if the main query is on the **Students** type, the **Person** navigation property will be available to get the **Person. FullName**. A requirement for using the properties is that the OData query must include an **$expand** parameter to expand the navigation property.

Example

Query to select the top 10 students and expand the Person property, selecting just the FullName:

http://<studentwebclientbaseurl>/ds/campusnexus/Students?$select=Id&$expand=Person($select=FullName)&$top=10

Within the workflow, the FullName can be accessed as **item.Person.FullName**.

Properties

ExecuteODataQuery Properties

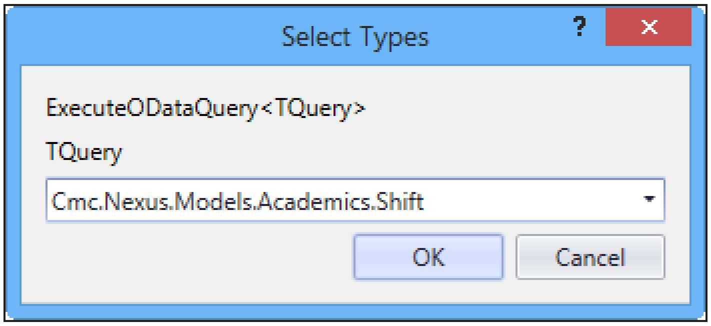
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
ItemCount	`OutArgument<Int32>`	No	This property can be used to capture the total count of items returned by the query.
QueryText	`InArgument<String>`	Yes	Specify the OData query string. Make sure the string is enclosed in double quotes.
RawODataResults	`OutArgument<String>`	No	This property can be used to provide the raw JSON string result.
Result	`OutArgument<TQuery>`	No	**This property is not supported.** Please use the ResultsCollection property.
ResultsCollection	`InOutArgument<IEnumerable<TQuery>>`	No	Use this property to save the collection of items to a variable. The variable type must be IEnumerable<TQuery>) and a default value must be defined.

ExecuteODataQuery<> Example

1. Drag the ExecuteODataQuery<> activity from the “Cmc.Core.Workflow.Activities” namespace into the Designer pane.

2. Select **Browse for Types ...** in the TQuery drop-down list.

3. Navigate to the **Cmc.Nexus.Models** namespace. For this example, we will use the **Shift** model in the Academics namespace. Selecting “Shift” means that the OData query will be written using “Shifts” as the primary model.



ExecuteODataQuery<TQuery>

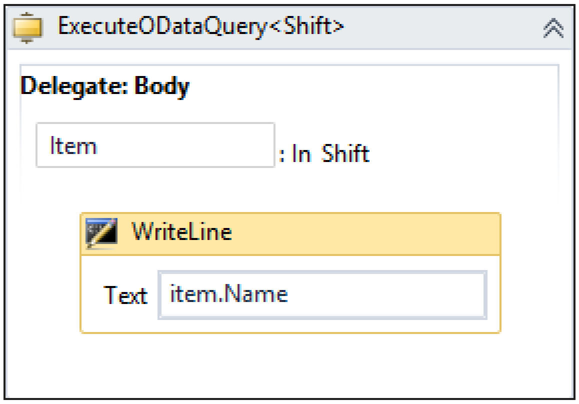
TQuery: Cmc.Nexus.Models.Academics.Shift

4. Click **OK** to close the Select Types dialog.

5. Paste the OData query into the **QueryText** field of the Properties pane. Make sure to add the string value surrounded by double quotes. In our example, the OData query is:

   "http://<studentwebclientbaseurl>/ds/campusnexus/Shifts?$orderby=Name"

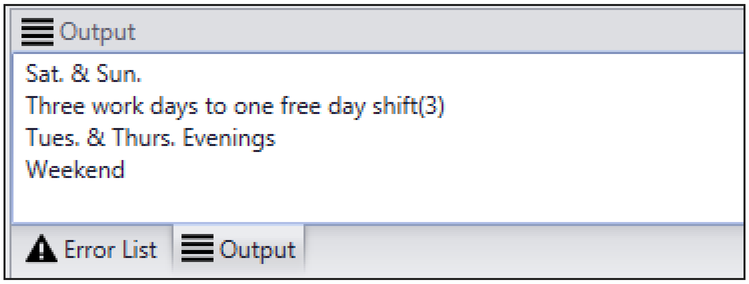
6. To capture the query results in the Output pane of Workflow Composer, drop a **WriteLine** activity into the body of the ExecuteODataQuery<> activity. Specify **item.Name** in the Text field of the WriteLine activity.



Item : In Shift

Text: item.Name

7. Click Run. The **Name** of each shift is written to the Output pane.

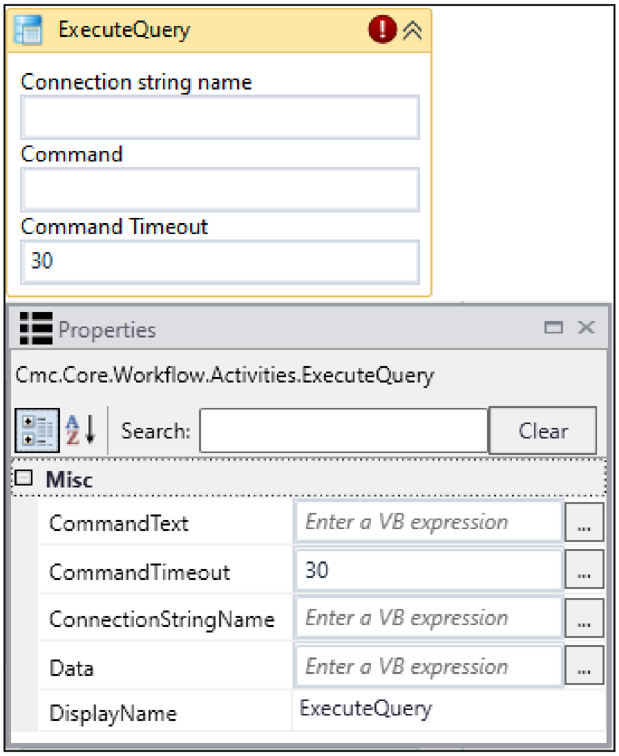


```
Sat. & Sun.
Three work days to one free day shift(3)
Tues. & Thurs. Evenings
Weekend
```

ExecuteQuery

The ExecuteQuery activity enables you to create workflows that perform SQL queries into an ADO.NET data source to return a result set of data from a given data source.

If the query result is not empty, the workflow can be programmed to iterate over the result set and execute logic for each data record by using a ForEach<T> activity.



Connection string name:

Command:

Command Timeout: 30

Cmc.Core.Workflow.Activities.ExecuteQuery

CommandText	Enter a VB expression
CommandTimeout	30
ConnectionStringName	Enter a VB expression
Data	Enter a VB expression
DisplayName	ExecuteQuery

In general, the connection strings used during workflow execution are retrieved from the web.config of the product that triggers workflow execution.

Only if you want to run a workflow with ExecuteDataReader, ExecuteNonQuery, or ExecuteQuery activity in test mode using the **Run** option in Workflow Composer, would you need to manually add the connection string to the Workflow Composer web.config file.

Properties

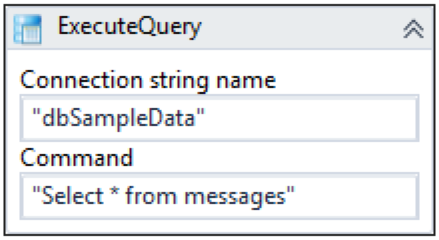
ExecuteQuery Properties

Property	Value	Required	Notes
CommandText	`InArgument<String>`	Yes	Enter a command that specifies the query to perform on the target data source and is expected to return a result set. *Example:* `"Select * from Messages"`

Property	Value	Required	Notes
CommandTimeout	`InArgument<Int32>`	No	You can adjust the CommandTimeout value if the activity needs to execute long-running SQL statements. The default and minimum command timeout is 30 seconds. The maximum is 1800 (30 minutes).
ConnectionStringName	`InArgument<String>`	Yes	Enter the name of a connection string that has been configured in the CONFIG file of the host application that is executing the workflows (see Connection Strings). If none is specified, this activity attempts to connect to a connection string named `DbConnection`. **Note**: Forms Builder 3.6 introduces the "CrmConnection" string in the web.config of Forms Renderer (see Renderer Connection Strings). If you have created workflows with ExecuteQuery activities, ensure that connection strings in the activities match the updated web.config of Forms Renderer.
Data	`OutArgument<Int32>`	No	The output argument contains the data returned by the query. It may return one or more System.Data.DataTable objects depending on the results of the query execution.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

ExecuteQuery Example 1

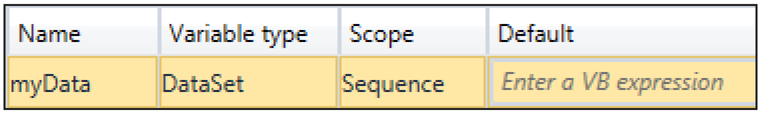
1. Open a workflow or create a new workflow.
2. Drag the **ExecuteQuery** activity into your workflow.
3. Specify the values for the input arguments or map them to workflow variables.



Connection string name: "dbSampleData"

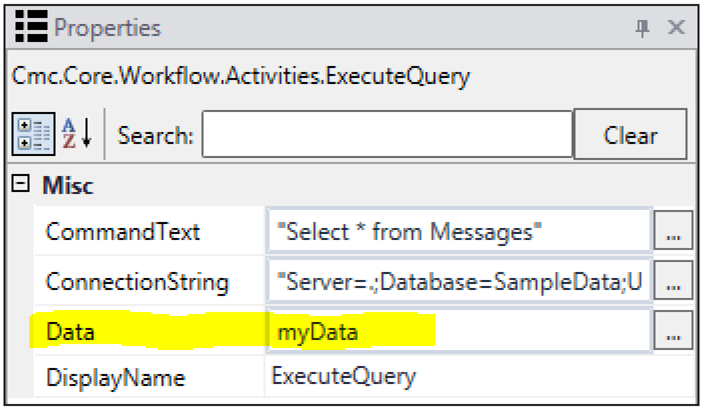
Command: "Select * from messages"

4. Create a workflow variable of data type **System.Data.DataSet** that will be mapped to the OutArgument of the query.



Name	Variable type	Scope	Default
myData	DataSet	Sequence	*Enter a VB expression*

5. Map the OutArgument named Data to the new workflow variable.



Cmc.Core.Workflow.Activities.ExecuteQuery

Misc

CommandText	"Select * from Messages"
ConnectionString	"Server=.;Database=SampleData;U
Data	myData
DisplayName	ExecuteQuery

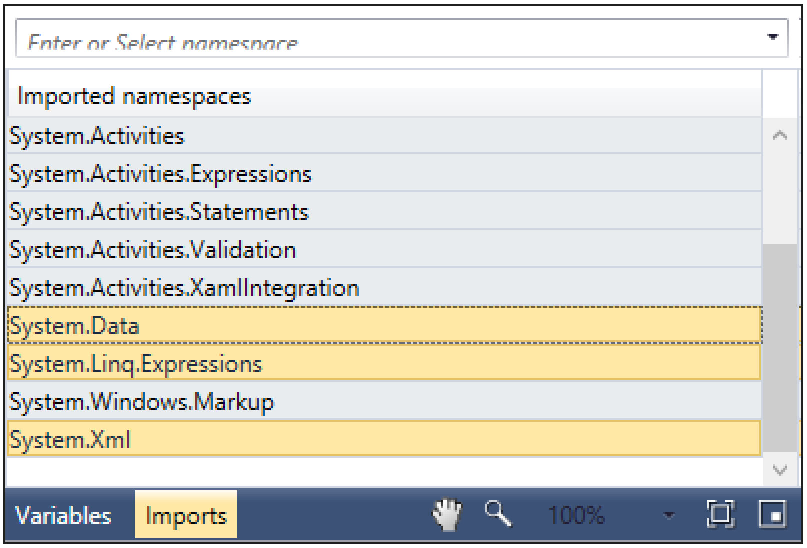
6. Import the following namespaces into the workflow:

   - System.Data
   - System.Linq.Expression
   - System.Xml

   These namespaces are needed to allow the ForEach<T> activity to easily iterate over the results in each System.Data.DataTable object returned.

   To import the namespaces:

   a. Click the **Imports** pane in the Workflow Designer.
   b. Click ▾ on the right side of the "Enter or Select namespace" field.
   c. Type the name of the namespace you want to import.
   d. **Select** the namespace and press **Enter**.



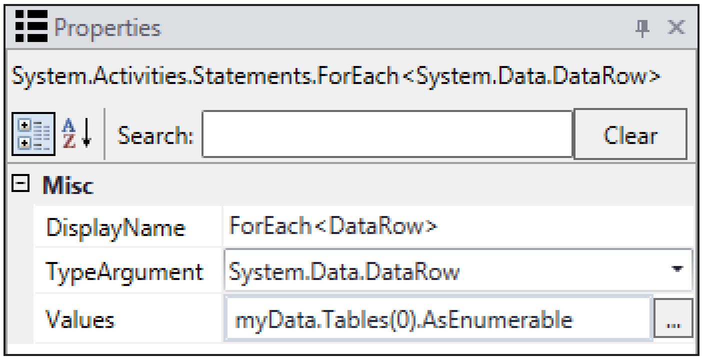
Imported namespaces

- System.Activities
- System.Activities.Expressions
- System.Activities.Statements
- System.Activities.Validation
- System.Activities.XamlIntegration
- System.Data
- System.Linq.Expressions
- System.Windows.Markup
- System.Xml

7. Add a **ForEach<T>** activity to your workflow.

   Configure **TypeArgument = System.Data.DataRow**.

   You can assign the **Values** variable to each DataTable returned as shown below.

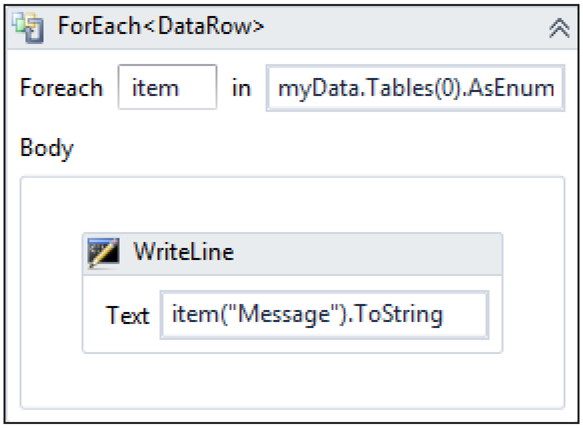


System.Activities.Statements.ForEach<System.Data.DataRow>

DisplayName	ForEach<DataRow>
TypeArgument	System.Data.DataRow
Values	myData.Tables(0).AsEnumerable

8. Configure the **ForEach<T>** activity to assign a **name** to each row as it iterates through the rows returned from the database.

   In the example shown here, each row is assigned the variable name of **item**. Access the values returned in each row by using the format: `item(“ColumnName”)`



9. Run the workflow.

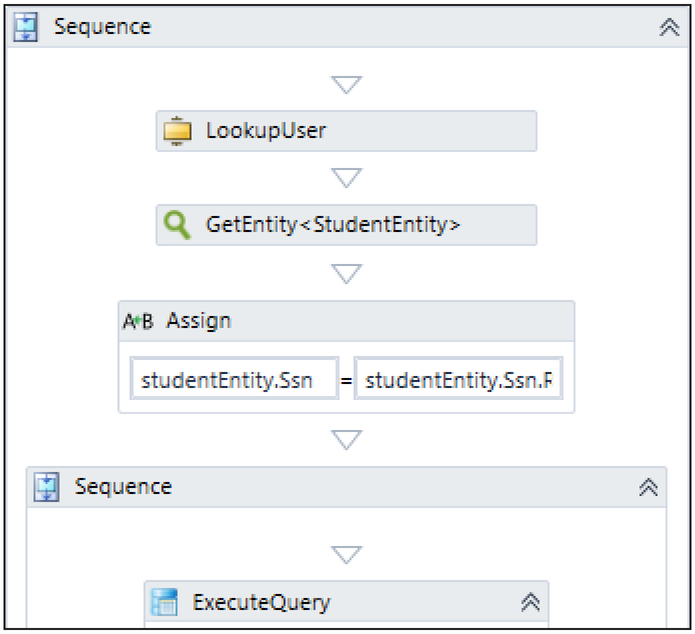
   Result:

   - If the query successfully connects to the data source, the activity populates your local variable with the rows returned by the query.
   - The ForEach<T> activity iterates over each row stored in the local variable. It executes the activities within the body of the ForEach activity per each row in the DataTable.

ExecuteQuery Example 2

The following example uses the ExecuteQuery activity in the context of a Forms Builder sequence. In the first form, the ExecuteQuery activity queries the database for a student's registration bill details by term and displays the data in a grid where each row represents a term.

1. The LookupUser activity captures the UserName from the formInstance.UserName argument and returns the SyStudentID value.
2. The GetEntity activity takes the SyStudentID and returns the studentEntity.
3. The Assign activity assigns the value `studentEntity.Ssn.Remove(1,7)` to `studentEntity.Ssn`. This formats the SSN to display only the last 4 digits. It starts at 1 and removes 7 digits. This includes the dashes (`111-11-1111`). So that leaves the last 4 of the SSN.

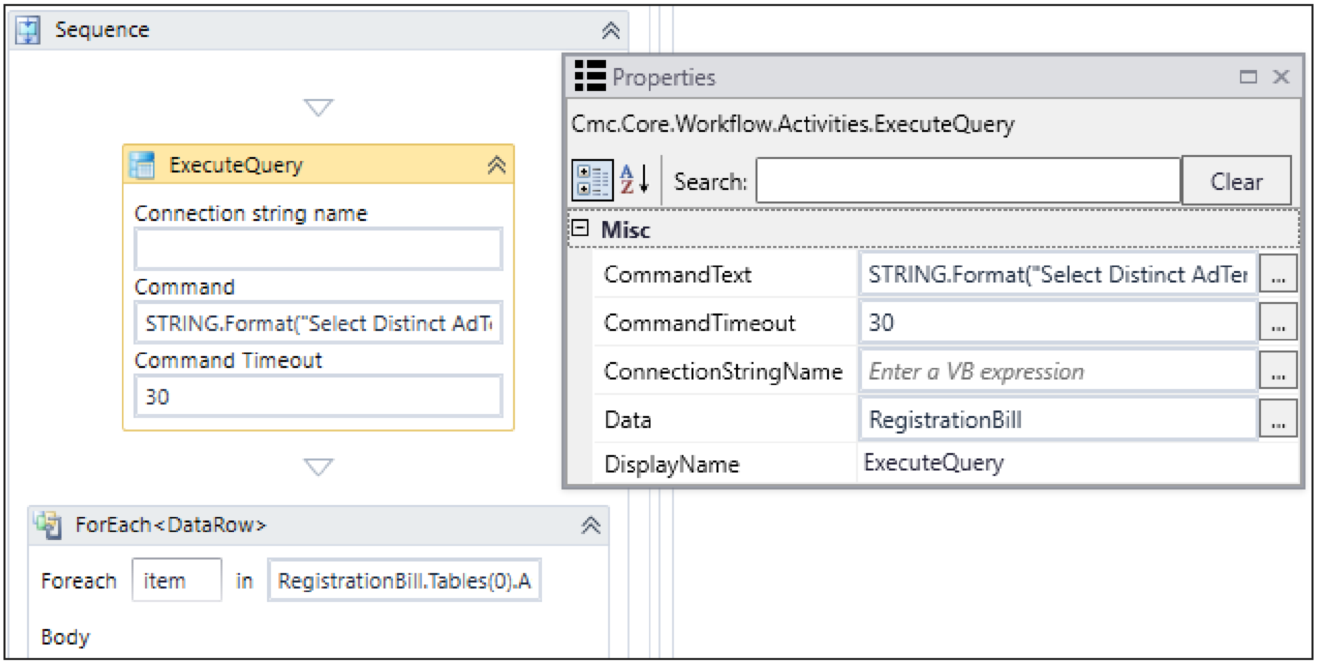


studentEntity.Ssn = studentEntity.Ssn.F

4. The ExecuteQuery activity queries the Anthology Student database for the student's registration billl details using the following SQL statement:

   STRING.Format("Select Distinct AdTerm.Descrip AS termSelect, SaTrans.AdTermId AS termSelectID from SaTrans join AdTerm on SaTrans.AdTermId = AdTerm.AdTermId WHERE SyStudentID = {0}",studententity.Id)

   The ExecuteQuery activity stores the retrieved data in a variable named "RegistrationBill". The variable type is a DataSet.



Connection string name:
Command: STRING.Format("Select Distinct AdT
Command Timeout: 30

Cmc.Core.Workflow.Activities.ExecuteQuery

CommandText	STRING.Format("Select Distinct AdTer
CommandTimeout	30
ConnectionStringName	Enter a VB expression
Data	RegistrationBill
DisplayName	ExecuteQuery

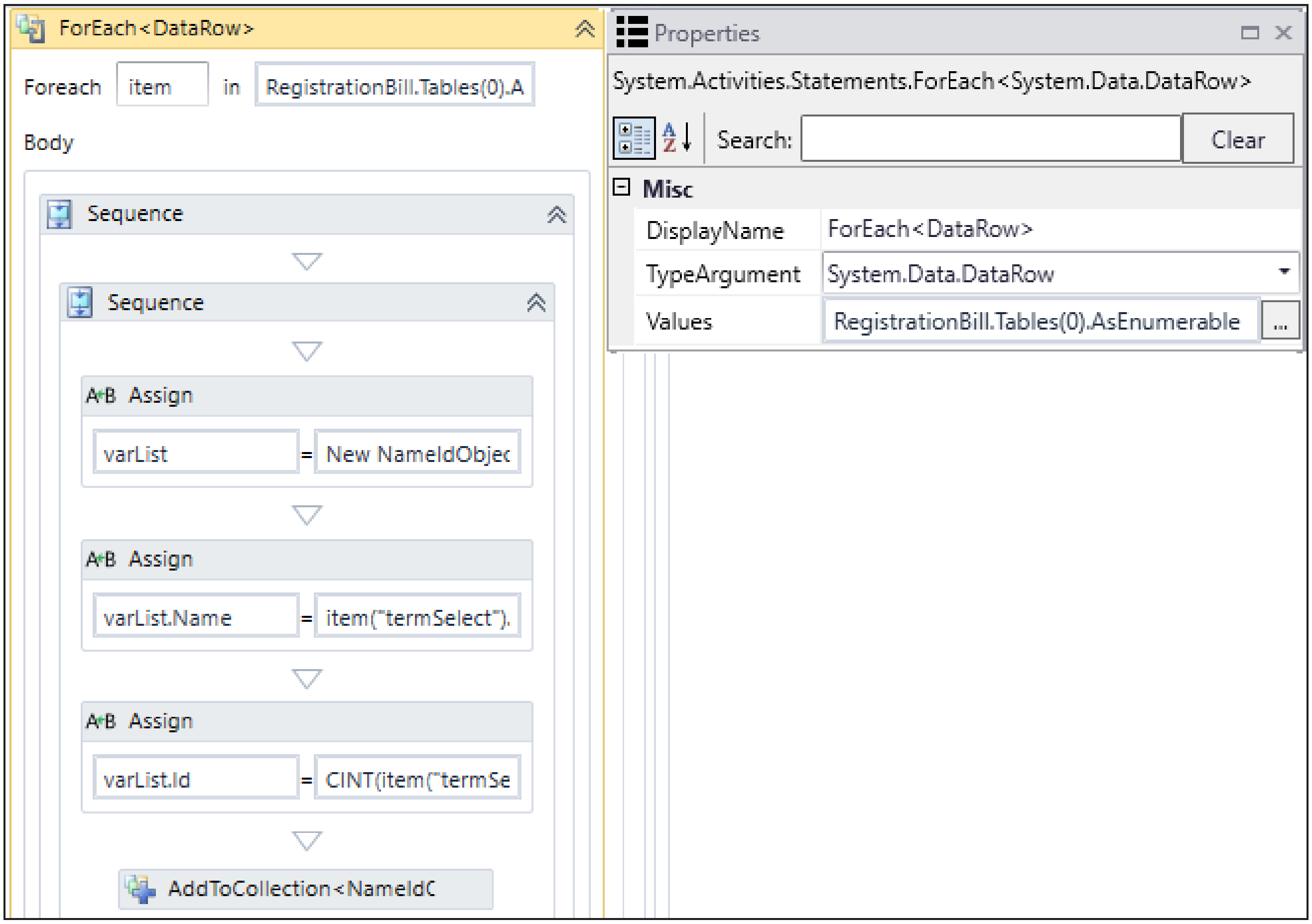
Foreach item in RegistrationBill.Tables(0).A
Body

5. Next, a ForEach activity parses the data output from the ExecuteQuery using the value **RegistrationBill.Tables(0).AsEnumerable** with the TypeArgument **System.Data.DataRow**.

   We are basically using the returned RegistrationBill DataSet, and in the ForEach activity we are looping through each row and then doing something with the data (e.g., assigning values).

6. The Body section of the ForEach activity includes three Assign activities that assign the following values to variables:

Variable	Value
varList	New NameIdObject
varListId	item("termSelect").ToString
varListName	CINT(item("termSelectId"))



Foreach item in RegistrationBill.Tables(0).A

varList = New NameIdObjec

varList.Name = item("termSelect").

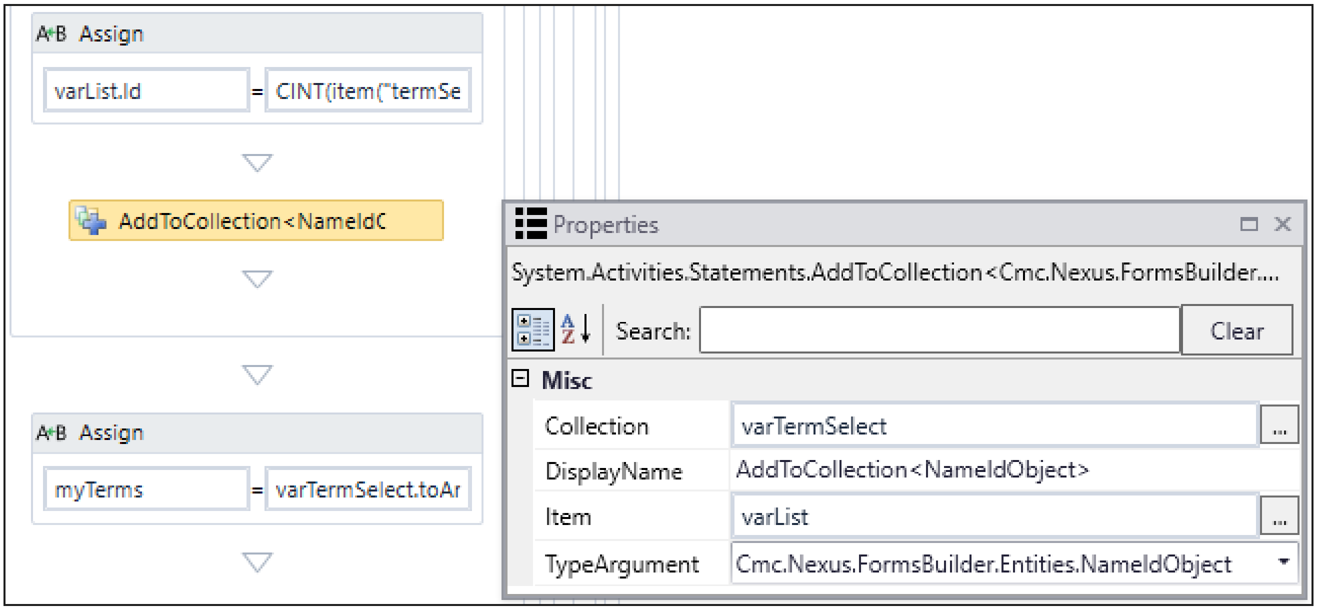
varList.Id = CINT(item("termSe

System.Activities.Statements.ForEach<System.Data.DataRow>

DisplayName	ForEach<DataRow>
TypeArgument	System.Data.DataRow
Values	RegistrationBill.Tables(0).AsEnumerable

7. The AddToCollection activity associates the varList variable with the NamedIdObject.

8. The Assign activity below the AddToCollection activity assigns the value **varTermSelect.toArray** to the **myTerms** argument. The value of this argument will be passed back to the form sequence and displayed in a grid row on the form.



A+B Assign
varList.Id = CINT(item("termSe
AddToCollection<NameIdC
A+B Assign
myTerms = varTermSelect.toAr
Properties
System.Activities.Statements.AddToCollection<Cmc.Nexus.FormsBuilder....
Search:
Clear
Misc

Collection	varTermSelect
DisplayName	AddToCollection<NameIdObject>
Item	varList
TypeArgument	Cmc.Nexus.FormsBuilder.Entities.NameIdObject

GetServiceInstance<>

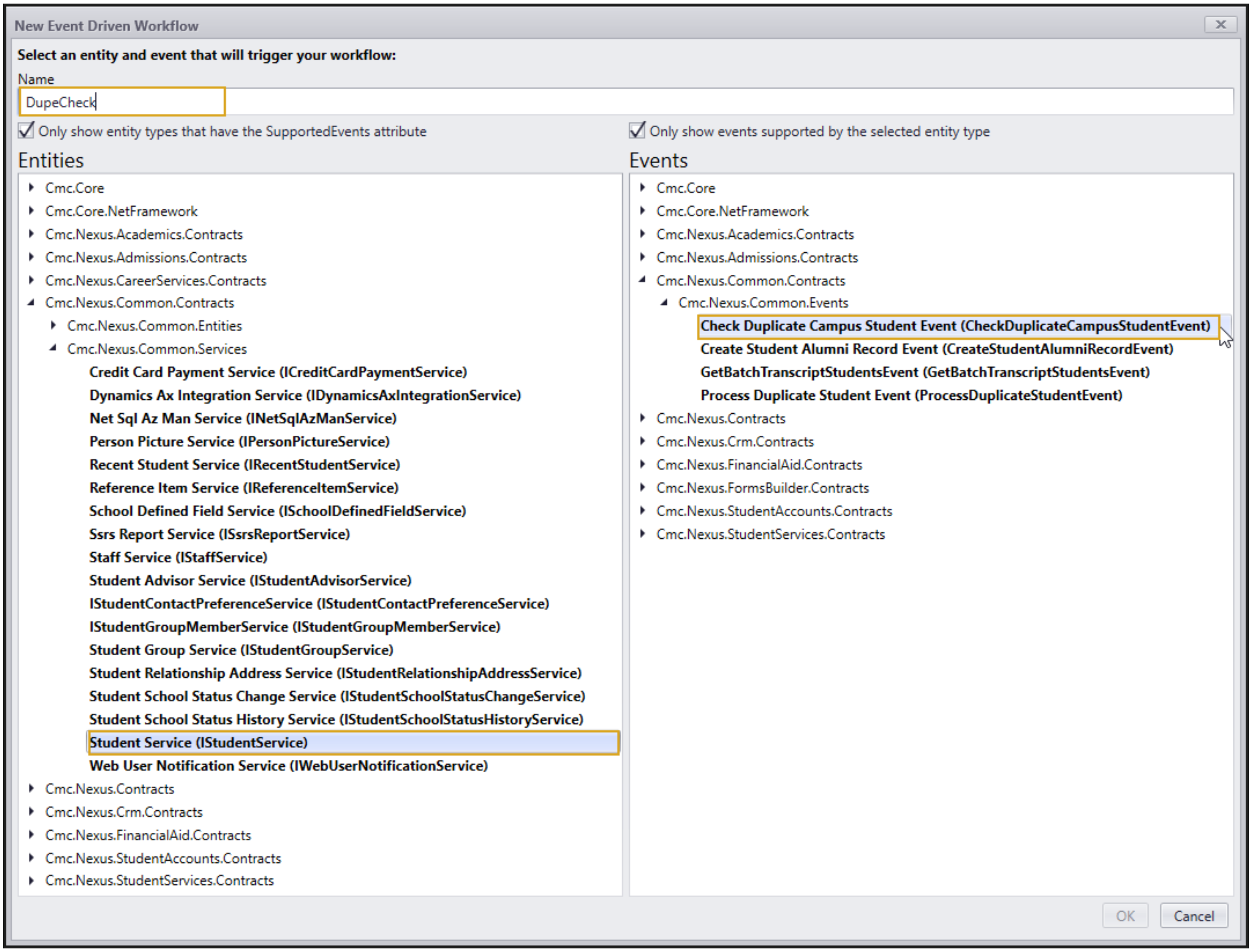
The GetServiceInstance activity retrieves an instance of a service from the service locator and provides the capability to execute service operations within the Anthology Student service suite.

The services and methods are documented in the Object Library. The Object Library is provided in compiled HTML (CHM) format and can be downloaded locally. Log on to https://www.mycampusinsight.com/Documentation-Center/Help/Help_Home/Content/helphome.htm and select **APIs** > **Object Library** > **Command Model** and **Query Model**.

The following operations invoked using the GetServiceInstance activity have gone through additional testing and have been proved out for use from Workflow Composer, but all methods are called internally by the application and should work.

- IStudentService - Check Duplicate Campus Student
- IStudentAccountTransactionService - Post Account Transaction Payment

To find available services, in Workflow Composer click **New Event Workflow** and select a service in the **Entities** pane. The bold text in the **Events** pane indicates the events supported by a selected service, for example, CheckDuplicateCampusStudentEvent. When called via the GetServiceInstance activity, the CheckDuplicateCampusStudentEvent becomes the CheckDuplicateCampusStudent method call on the iStudentService in the workflow. This screenshot shows how to find all available service methods.



Select an entity and event that will trigger your workflow:

Name: DupeCheck

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

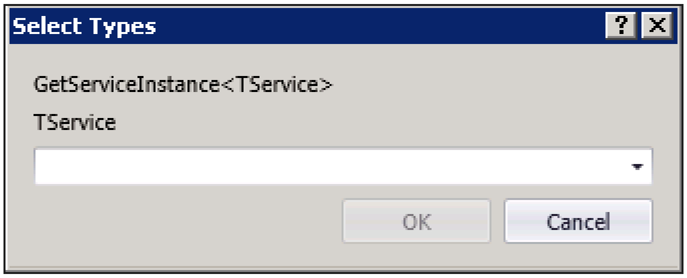
Entities

```
▸ Cmc.Core
▸ Cmc.Core.NetFramework
▸ Cmc.Nexus.Academics.Contracts
▸ Cmc.Nexus.Admissions.Contracts
▸ Cmc.Nexus.CareerServices.Contracts
◢ Cmc.Nexus.Common.Contracts
    ▸ Cmc.Nexus.Common.Entities
    ◢ Cmc.Nexus.Common.Services
        Credit Card Payment Service (ICreditCardPaymentService)
        Dynamics Ax Integration Service (IDynamicsAxIntegrationService)
        Net Sql Az Man Service (INetSqlAzManService)
        Person Picture Service (IPersonPictureService)
        Recent Student Service (IRecentStudentService)
        Reference Item Service (IReferenceItemService)
        School Defined Field Service (ISchoolDefinedFieldService)
        Ssrs Report Service (ISsrsReportService)
        Staff Service (IStaffService)
        Student Advisor Service (IStudentAdvisorService)
        IStudentContactPreferenceService (IStudentContactPreferenceService)
        IStudentGroupMemberService (IStudentGroupMemberService)
        Student Group Service (IStudentGroupService)
        Student Relationship Address Service (IStudentRelationshipAddressService)
        Student School Status Change Service (IStudentSchoolStatusChangeService)
        Student School Status History Service (IStudentSchoolStatusHistoryService)
        Student Service (IStudentService)
        Web User Notification Service (IWebUserNotificationService)
▸ Cmc.Nexus.Contracts
▸ Cmc.Nexus.Crm.Contracts
▸ Cmc.Nexus.FinancialAid.Contracts
▸ Cmc.Nexus.StudentAccounts.Contracts
▸ Cmc.Nexus.StudentServices.Contracts
```

Events

```
▸ Cmc.Core
▸ Cmc.Core.NetFramework
▸ Cmc.Nexus.Academics.Contracts
▸ Cmc.Nexus.Admissions.Contracts
◢ Cmc.Nexus.Common.Contracts
    ◢ Cmc.Nexus.Common.Events
        Check Duplicate Campus Student Event (CheckDuplicateCampusStudentEvent)
        Create Student Alumni Record Event (CreateStudentAlumniRecordEvent)
        GetBatchTranscriptStudentsEvent (GetBatchTranscriptStudentsEvent)
        Process Duplicate Student Event (ProcessDuplicateStudentEvent)
▸ Cmc.Nexus.Contracts
▸ Cmc.Nexus.Crm.Contracts
▸ Cmc.Nexus.FinancialAid.Contracts
▸ Cmc.Nexus.FormsBuilder.Contracts
▸ Cmc.Nexus.StudentAccounts.Contracts
▸ Cmc.Nexus.StudentServices.Contracts
```

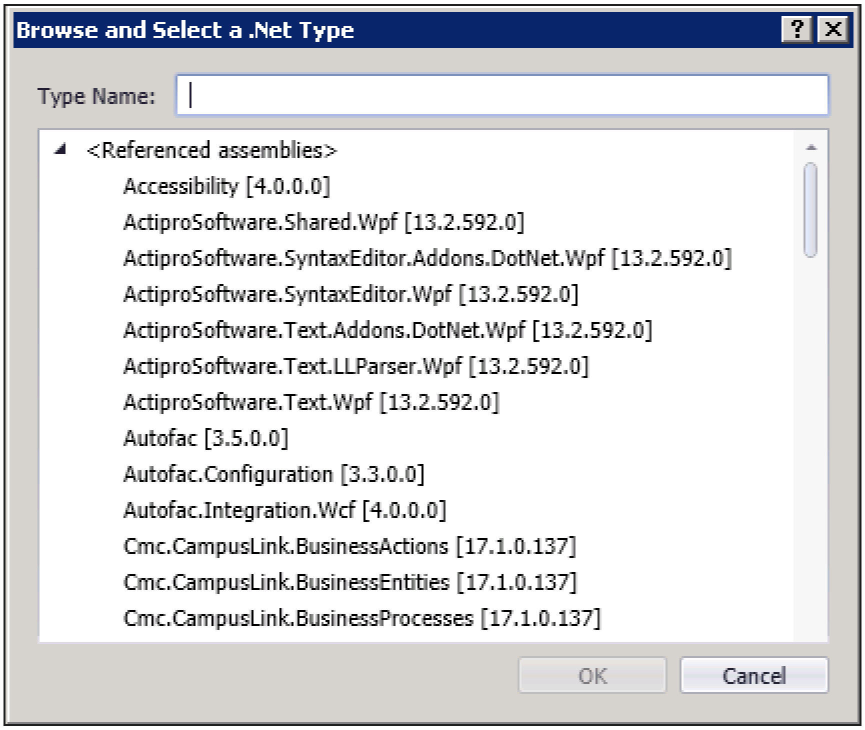
When you drag the GetServiceInstance activity into the Designer window, you are prompted to select the service type (TService). Click ▾.



GetServiceInstance<TService>

TService

When you select the 'Browse for Type' option, the list of assemblies and associated services is displayed. Find and select the service and click **OK**.

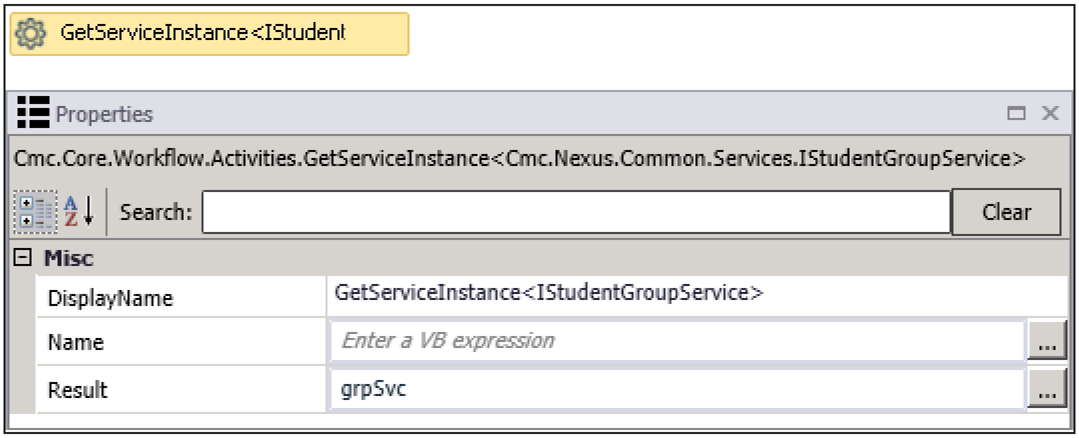


Browse and Select a .Net Type

Type Name:

<Referenced assemblies>
- Accessibility [4.0.0.0]
- ActiproSoftware.Shared.Wpf [13.2.592.0]
- ActiproSoftware.SyntaxEditor.Addons.DotNet.Wpf [13.2.592.0]
- ActiproSoftware.SyntaxEditor.Wpf [13.2.592.0]
- ActiproSoftware.Text.Addons.DotNet.Wpf [13.2.592.0]
- ActiproSoftware.Text.LLParser.Wpf [13.2.592.0]
- ActiproSoftware.Text.Wpf [13.2.592.0]
- Autofac [3.5.0.0]
- Autofac.Configuration [3.3.0.0]
- Autofac.Integration.Wcf [4.0.0.0]
- Cmc.CampusLink.BusinessActions [17.1.0.137]
- Cmc.CampusLink.BusinessEntities [17.1.0.137]
- Cmc.CampusLink.BusinessProcesses [17.1.0.137]

After you have selected a service, the name of the service is inserted into the DisplayName field, e.g., GetServiceInstance<IStudentGroupService>. Proceed to specify the Name and Result.



GetServiceInstance<IStudent

Properties

Cmc.Core.Workflow.Activities.GetServiceInstance<Cmc.Nexus.Common.Services.IStudentGroupService>

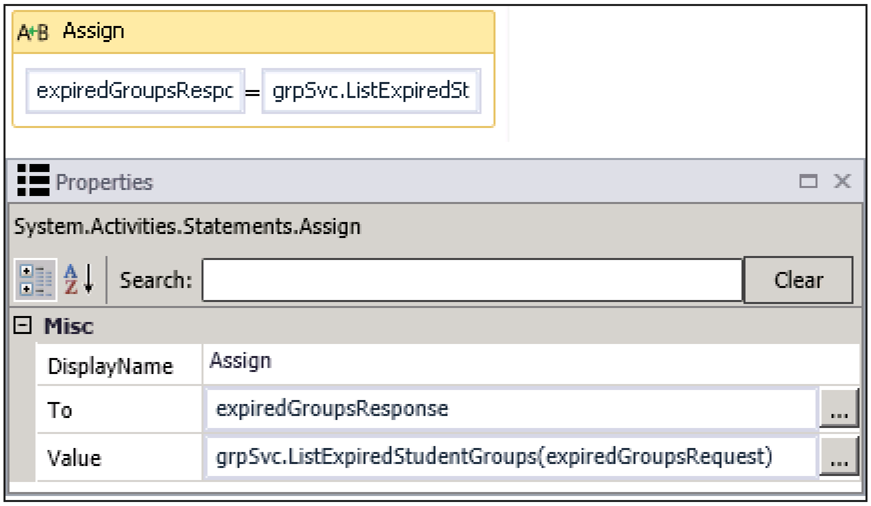
Search:

Misc

DisplayName	GetServiceInstance<IStudentGroupService>
Name	*Enter a VB expression*
Result	grpSvc

In the example above, the GetServiceInstance activity is associated with a variable (grpSvc) that detects the IStudentGroupService.

The workflow sequence continues with an Assign activity that assigns the variable from GetServiceInstance activity to the "expiredGroupsResponse" value. The Assign activity invokes the ListExpiredStudentGroups method of the iStudentGroupService. See IStudentGroupService Methods in the Anthology Student Object Library. The end result is that the workflow captures all expired student groups.

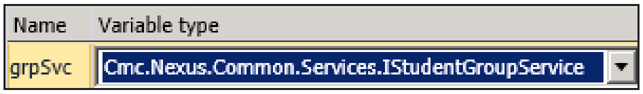


System.Activities.Statements.Assign

DisplayName	Assign
To	expiredGroupsResponse
Value	grpSvc.ListExpiredStudentGroups(expiredGroupsRequest)

Properties

GetServiceInstance<> Properties

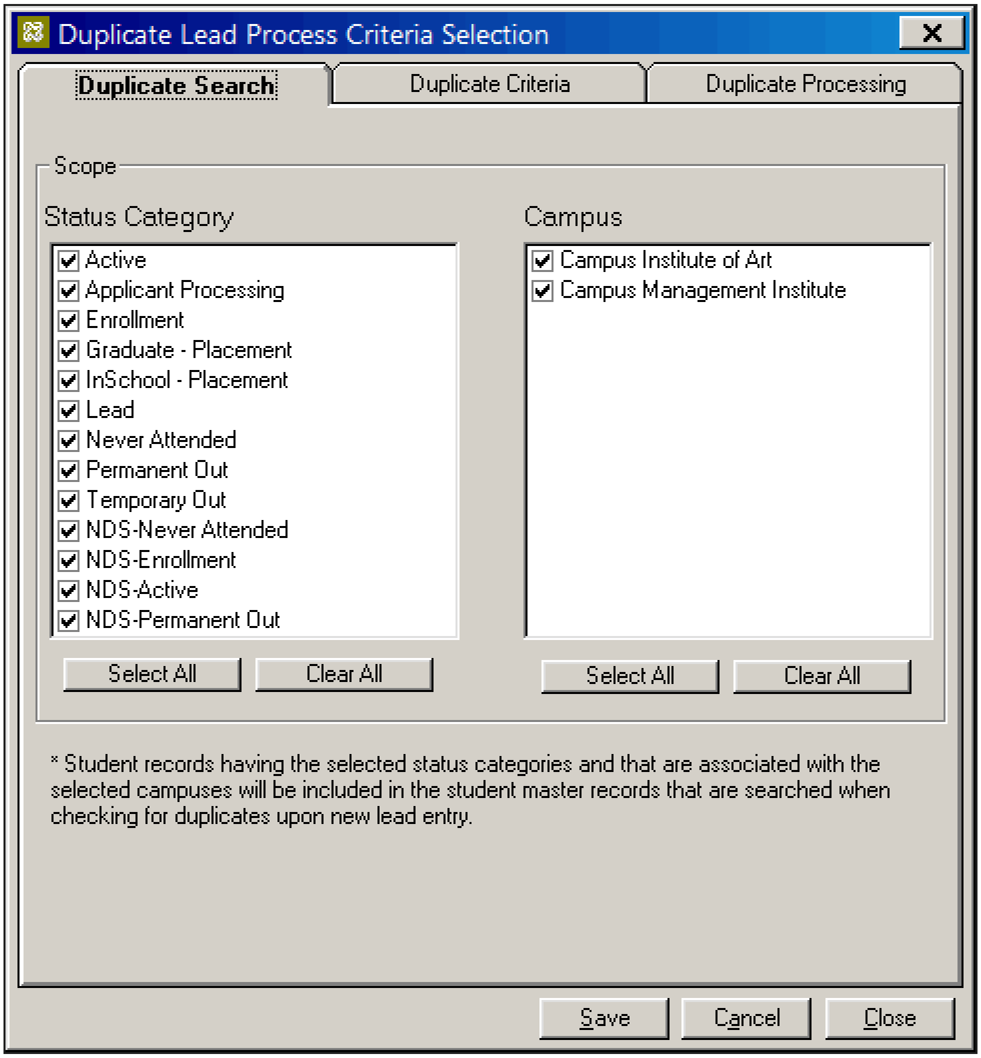
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Name	`InArgument<String>`	No	Specify a name for the service using a VB expression or variable.
Result	`OutArgument<Service>`	No	The service retrieved by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to the service that matches the previously selected service type, for example, **Cmc.Nexus.Common.Services.IStudentGroupService** and click **OK**.  Name: grpSvc Variable type: Cmc.Nexus.Common.Services.IStudentGroupService

IStudentService - Check Duplicate Campus Student

You can use the GetServiceInstance<> activity to invoke the IStudentService method. One of the operations that can be executed using the IStudentService method is the CheckDuplicateCampusStudent operation. The response will indicate whether one or more potential duplicate students are found or not.

Duplicate Lead Process Configuration

In the Anthology Student desktop client, navigate to **Setup** > **Campus Locations** > select a campus > **Add/Edit** (button) > **Allow...** (tab). Click **Duplicate Lead Process Configuration** button and review the settings on the Duplicate Search, Duplicate Criteria, and Duplicate Processing tabs.



Scope

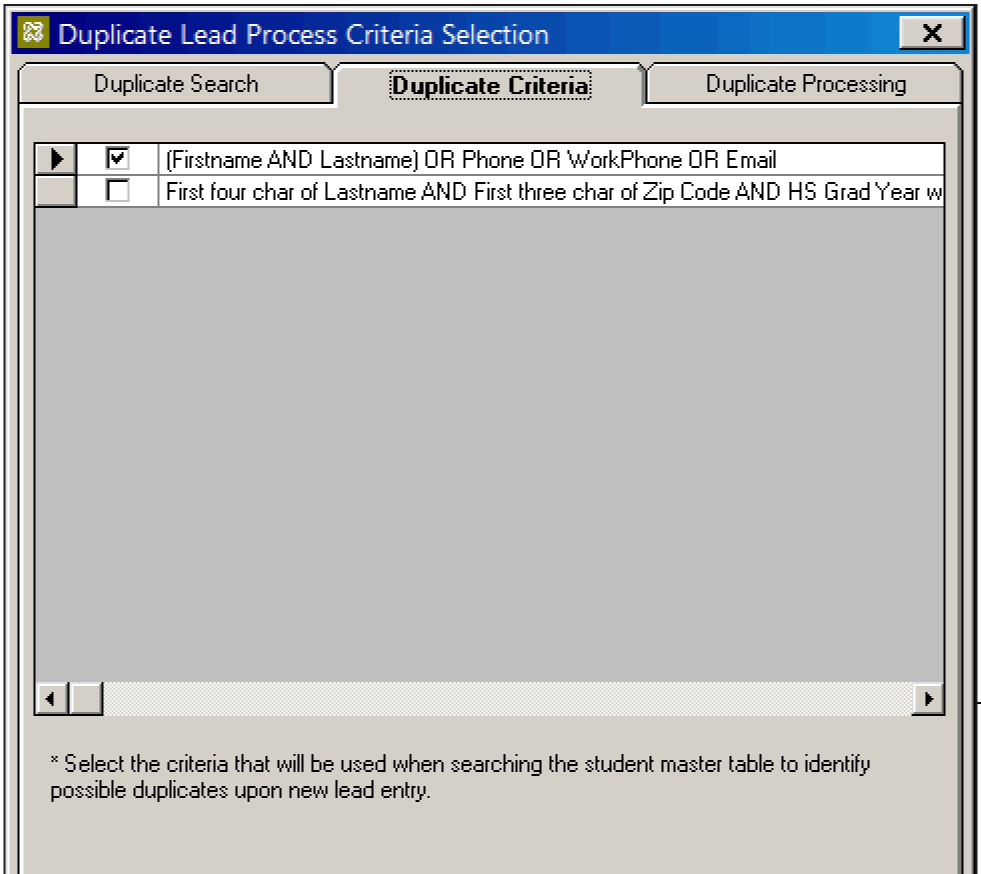
Status Category

- ☑ Active
- ☑ Applicant Processing
- ☑ Enrollment
- ☑ Graduate - Placement
- ☑ InSchool - Placement
- ☑ Lead
- ☑ Never Attended
- ☑ Permanent Out
- ☑ Temporary Out
- ☑ NDS-Never Attended
- ☑ NDS-Enrollment
- ☑ NDS-Active
- ☑ NDS-Permanent Out

Campus

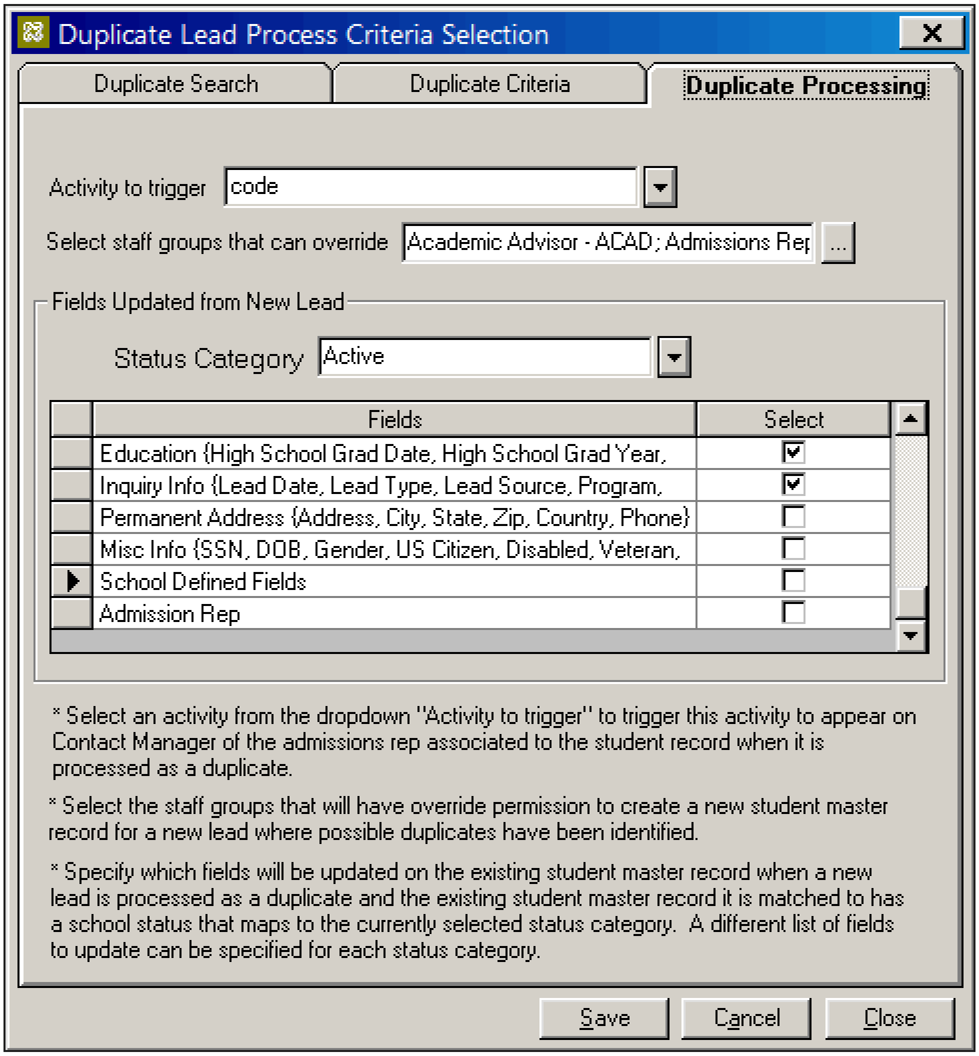
- ☑ Campus Institute of Art
- ☑ Campus Management Institute

* Student records having the selected status categories and that are associated with the selected campuses will be included in the student master records that are searched when checking for duplicates upon new lead entry.



- ☑ (Firstname AND Lastname) OR Phone OR WorkPhone OR Email
- □ First four char of Lastname AND First three char of Zip Code AND HS Grad Year w

* Select the criteria that will be used when searching the student master table to identify possible duplicates upon new lead entry.



Activity to trigger: code

Select staff groups that can override: Academic Advisor - ACAD; Admissions Rep

Fields Updated from New Lead

Status Category: Active

Fields	Select
Education {High School Grad Date, High School Grad Year,	☑
Inquiry Info {Lead Date, Lead Type, Lead Source, Program,	☑
Permanent Address {Address, City, State, Zip, Country, Phone}	☐
Misc Info {SSN, DOB, Gender, US Citizen, Disabled, Veteran,	☐
School Defined Fields	☐
Admission Rep	☐

* Select an activity from the dropdown "Activity to trigger" to trigger this activity to appear on Contact Manager of the admissions rep associated to the student record when it is processed as a duplicate.

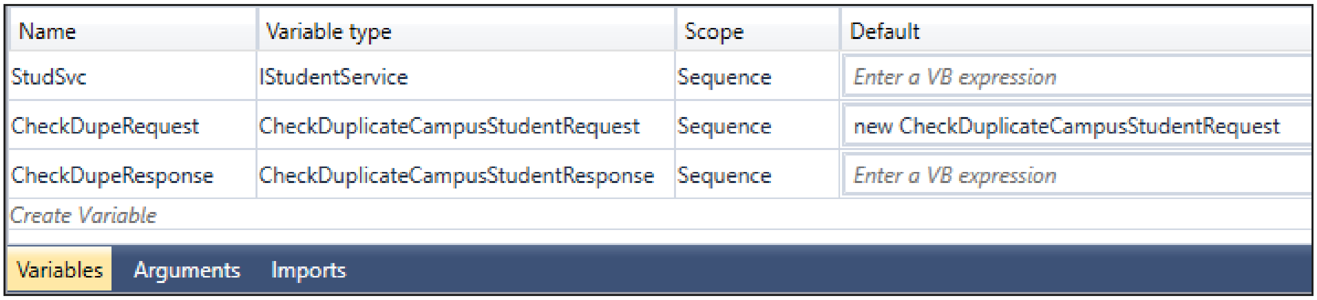
* Select the staff groups that will have override permission to create a new student master record for a new lead where possible duplicates have been identified.

* Specify which fields will be updated on the existing student master record when a new lead is processed as a duplicate and the existing student master record it is matched to has a school status that maps to the currently selected status category. A different list of fields to update can be specified for each status category.

The duplicate search process is based on the criteria settings against records for the configured status categories and campuses.

Workflow Example

1. In Workflow Composer, create the following **Variables**. Be sure to use the indicated variable types and defaults.

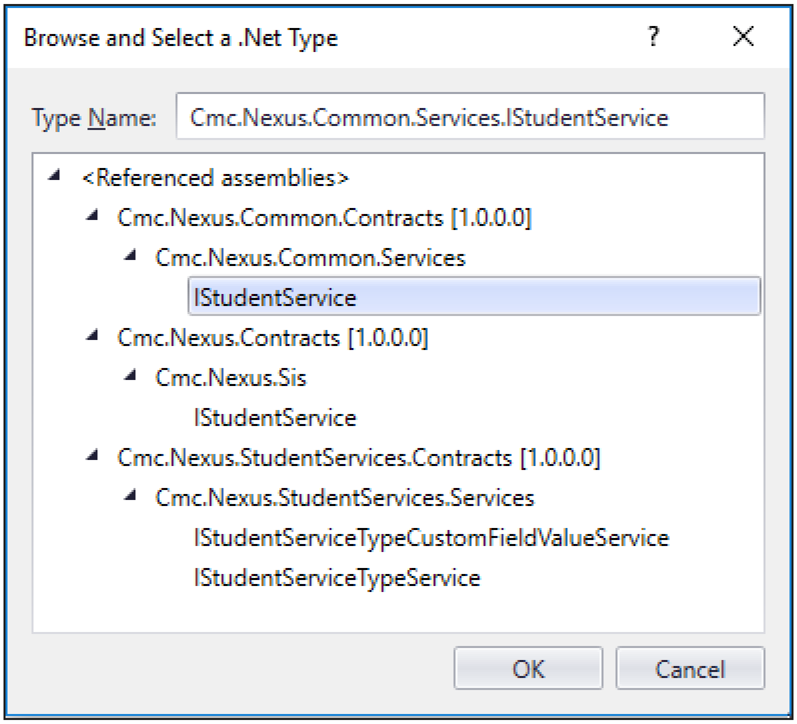


Name	Variable type	Scope	Default
StudSvc	IStudentService	Sequence	*Enter a VB expression*
CheckDupeRequest	CheckDuplicateCampusStudentRequest	Sequence	new CheckDuplicateCampusStudentRequest
CheckDupeResponse	CheckDuplicateCampusStudentResponse	Sequence	*Enter a VB expression*

Create Variable

2. Drag the **GetServiceInstance** activity into the workflow, click ▾, and select **Browse for Type**.

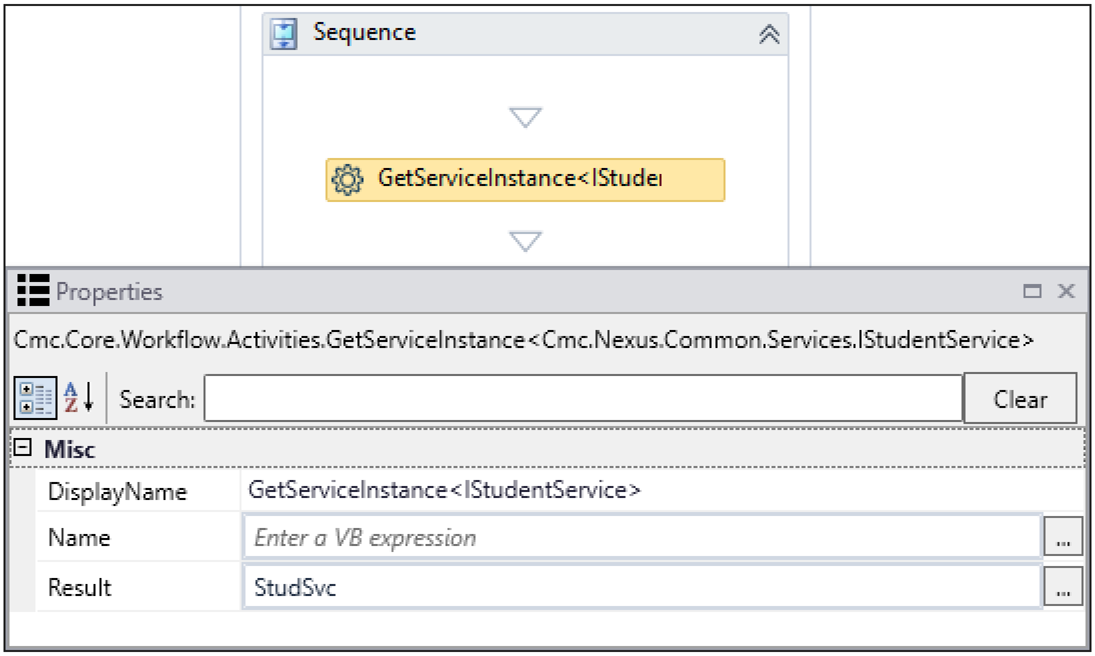
   Select **IStudentService** and click **OK**.



Type Name: Cmc.Nexus.Common.Services.IStudentService

- <Referenced assemblies>
  - Cmc.Nexus.Common.Contracts [1.0.0.0]
    - Cmc.Nexus.Common.Services
      - IStudentService
  - Cmc.Nexus.Contracts [1.0.0.0]
    - Cmc.Nexus.Sis
      - IStudentService
  - Cmc.Nexus.StudentServices.Contracts [1.0.0.0]
    - Cmc.Nexus.StudentServices.Services
      - IStudentServiceTypeCustomFieldValueService
      - IStudentServiceTypeService

3. In the **Result** field of the GetServiceInstance property window, specify the variable of type IStudentService created above.



Cmc.Core.Workflow.Activities.GetServiceInstance<Cmc.Nexus.Common.Services.IStudentService>

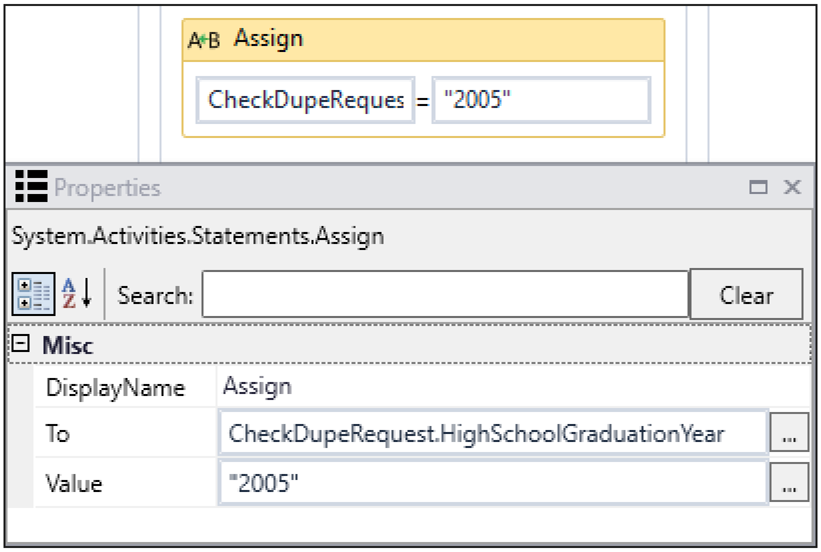
DisplayName	GetServiceInstance<IStudentService>
Name	*Enter a VB expression*
Result	StudSvc

4. Drag **Assign** activities into the workflow for each field you want to check for duplicates. The available duplicate check criteria include the following fields.

Duplicate Check Criteria

Student Entity Field	Assign Activity	
	To	Value
First Name	CheckDupeRequest.FirstName	Variable created in your workflow or hard-coded value
Last Name	CheckDupeRequest.LastName	
Phone	CheckDupeRequest.Phone	
Work Phone	CheckDupeRequest.WorkPhone	
Email	CheckDupeRequest.EMail	
ZIP Code	CheckDupeRequest.PostalCode	
HS Graduation Year	CheckDupeRequest.HighSchoolGraduationYear	
	CheckDupeRequest.IsDupNamePhoneCheckRequired	False

To check for duplicates on all of the fields listed above, an Assign activity is required for each field. You can choose to check for duplicates only on selected fields. The following is an example of Assign activity properties for the HS Graduation Year field, where "2005" is a hard-code a value.



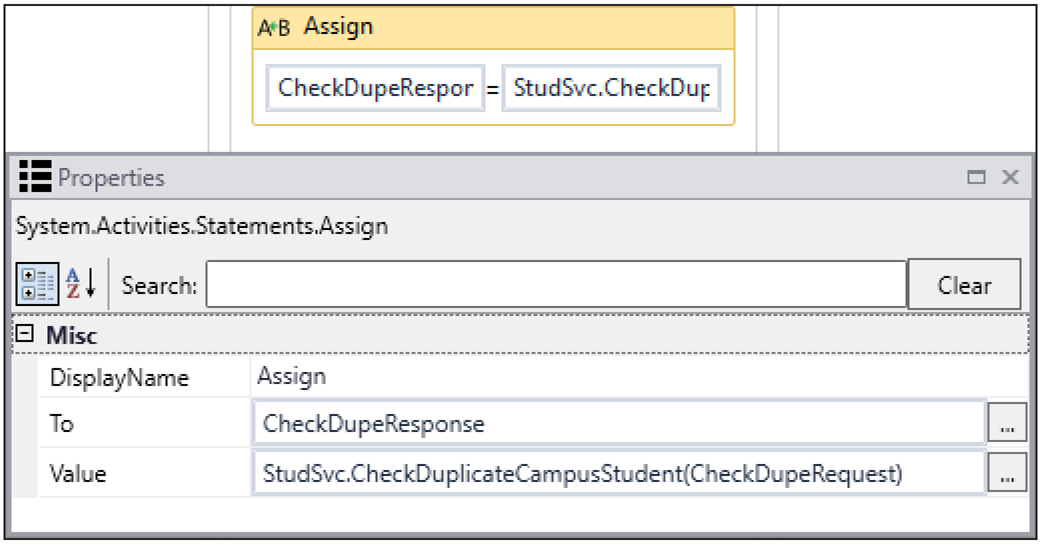
CheckDupeReques = "2005"

System.Activities.Statements.Assign

Misc

DisplayName	Assign
To	CheckDupeRequest.HighSchoolGraduationYear
Value	"2005"

5. Drag an **Assign** activity into the workflow to assign the response to the duplicate check.
   - In the "To" field specify: **CheckDupeResponse** (This is a variable created above.)
   - In the "Value" field specify: **StudSvc.CheckDuplicateCampusStudent(CheckDupeRequest)** (Where "StudSvc" is a variable created above.)



CheckDupeRespor = StudSvc.CheckDup

System.Activities.Statements.Assign

DisplayName	Assign
To	CheckDupeResponse
Value	StudSvc.CheckDuplicateCampusStudent(CheckDupeRequest)

6. To capture the result of the duplicate check, insert WriteLine, LogLine, or any other activities as appropriate for your workflow.

   As a quick way to determine if any duplicates found for further processing is to check the count of students returned, add an **If** activity and specify the following condition: **CheckDupeResponse.Students.Count > 0**

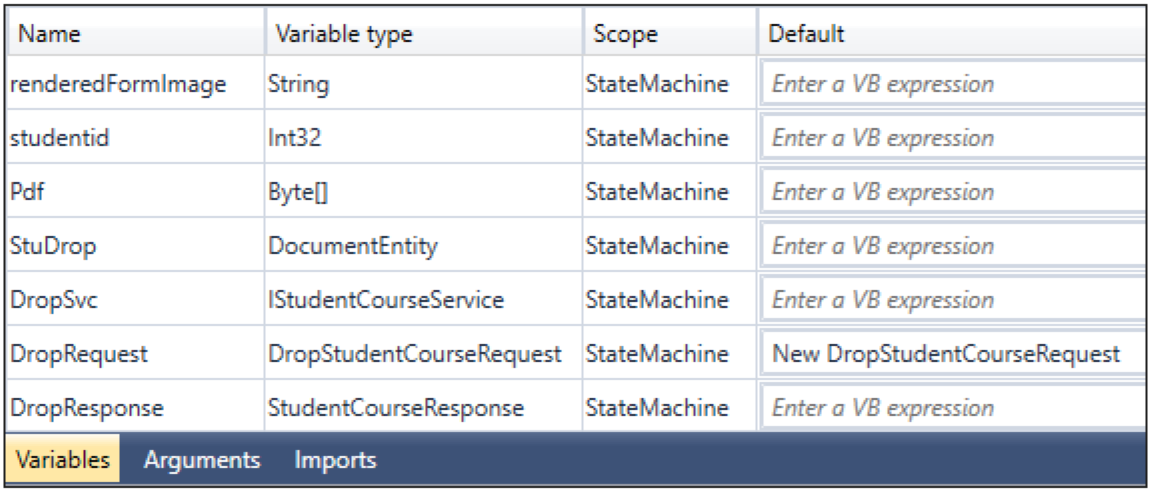
IStudentCourseService - Drop Course

You can use the GetServiceInstance<> activity to invoke the IStudentCourseService method to update a student course record in the Anthology Student database.

The workflow example below is associated with a Forms Builder form that allows students to submit an online form to drop a course to which they have previously been enrolled. The workflow creates and saves a document of the course drop request form and removes the dropped course from the student's course list.

Workflow Example

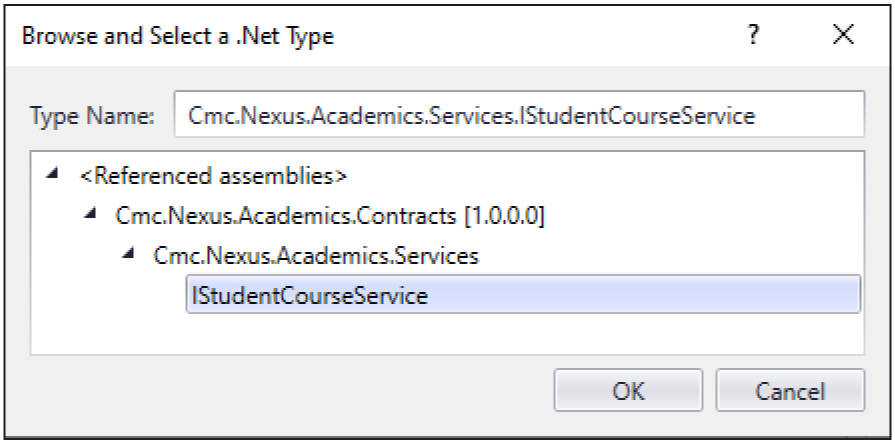
1. In Workflow Composer, create the following **Variables**. Be sure to use the indicated variable types and defaults.



Name	Variable type	Scope	Default
renderedFormImage	String	StateMachine	*Enter a VB expression*
studentid	Int32	StateMachine	*Enter a VB expression*
Pdf	Byte[]	StateMachine	*Enter a VB expression*
StuDrop	DocumentEntity	StateMachine	*Enter a VB expression*
DropSvc	IStudentCourseService	StateMachine	*Enter a VB expression*
DropRequest	DropStudentCourseRequest	StateMachine	New DropStudentCourseRequest
DropResponse	StudentCourseResponse	StateMachine	*Enter a VB expression*

Notes:

- For the **StuDrop** variable type, browse to Cmc.Nexus.Crm.Entities.DocumentEntity.
- For the **DropSvc** variable type, browse to Cmc.Nexus.Academics.Services.IStudentCourseService.

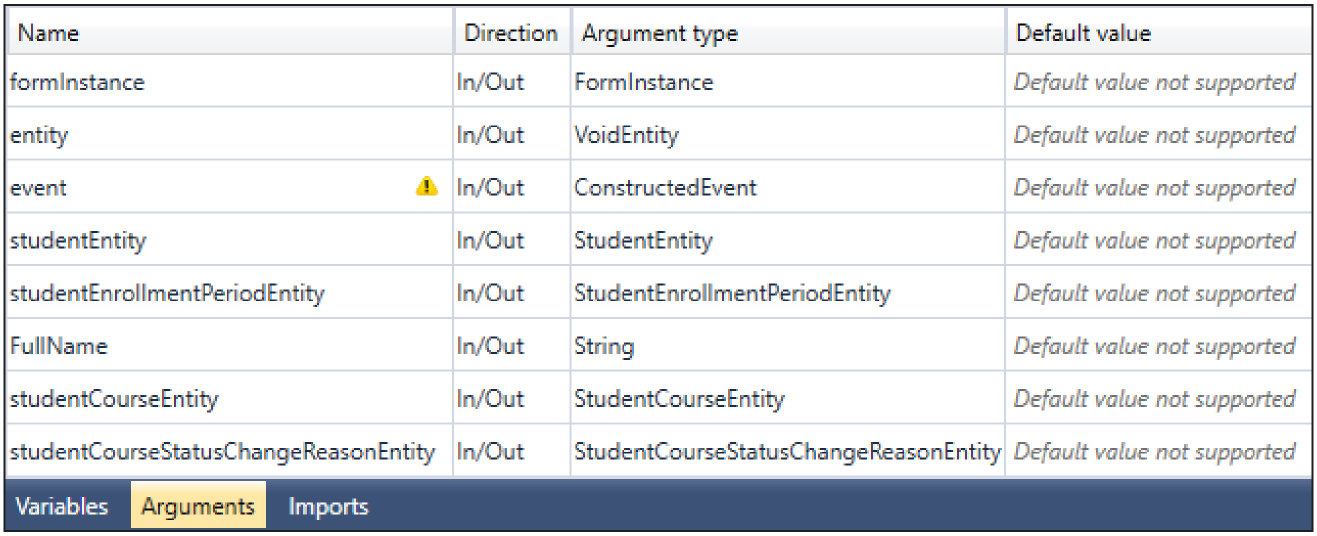


Type Name: Cmc.Nexus.Academics.Services.IStudentCourseService

<Referenced assemblies>
Cmc.Nexus.Academics.Contracts [1.0.0.0]
Cmc.Nexus.Academics.Services
IStudentCourseService

- For the **DropStudentCourseRequest** variable type, browse to Cmc.Nexus.Academics.Services.DropStudentCourseRequest.
- For the **StudentCourseResponse** variable type, browse to Cmc.Nexus.Academics.Services.StudentCourseResponse.

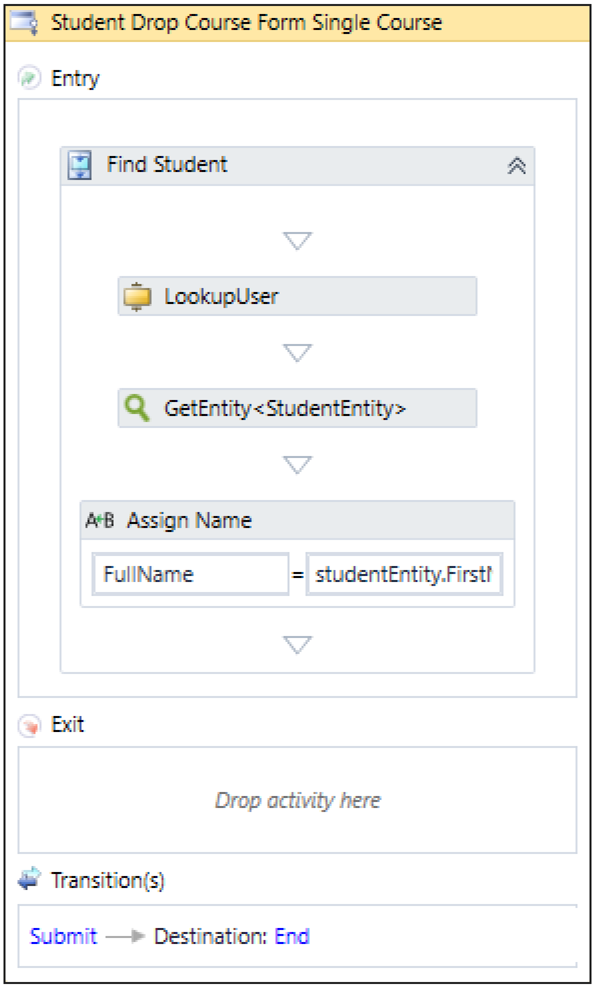
2. To exchange data with Forms Builder, the workflow uses the following arguments:



Name	Direction	Argument type	Default value
formInstance	In/Out	FormInstance	*Default value not supported*
entity	In/Out	VoidEntity	*Default value not supported*
event ⚠	In/Out	ConstructedEvent	*Default value not supported*
studentEntity	In/Out	StudentEntity	*Default value not supported*
studentEnrollmentPeriodEntity	In/Out	StudentEnrollmentPeriodEntity	*Default value not supported*
FullName	In/Out	String	*Default value not supported*
studentCourseEntity	In/Out	StudentCourseEntity	*Default value not supported*
studentCourseStatusChangeReasonEntity	In/Out	StudentCourseStatusChangeReasonEntity	*Default value not supported*

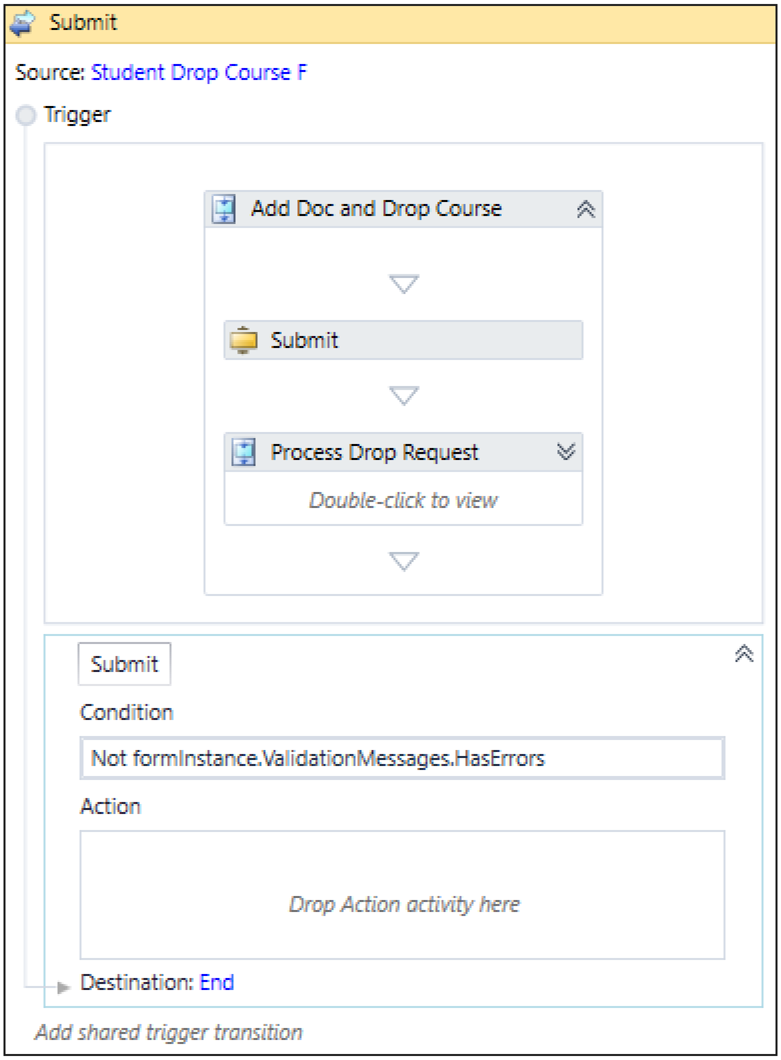
3. The first state/form finds the student record and assigns the student's name:

   - LookupUser determines **studentid** value.
   - GetEntity<StudentEntity> uses the **studentid** and returns the **studentEntity**.
   - Assign Name assigns **studentEntity.FirstName + " " + studentEntity.LastName** to the **Fullname** argument.



FullName = studentEntity.FirstI

4. The transition contains a WaitForFormBookmark activity (labeled "Submit"), a sequence with a number of activities to process the course drop request (see next step), and the validation condition **Not formInstance.ValidationMessages.HasErrors**.



Source: Student Drop Course F

Condition

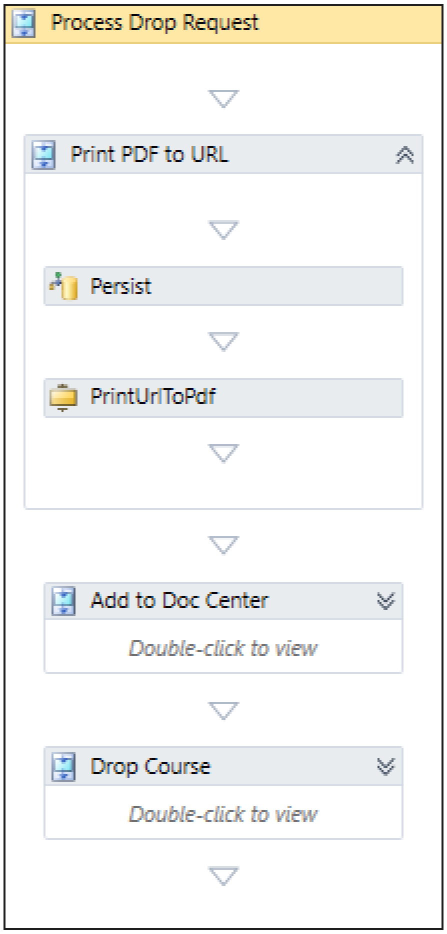
Not formInstance.ValidationMessages.HasErrors

Action

Destination: End

5. The "Process Drop Request" sequence creates a file of the course drop request form submitted by the student.

   - The **Persist** activity precedes the PrintUrlToPdf activity to explicitly request that the workflow persists its data to a file.
   - The PrintUrlToPdf activity creates a file named **Pdf** (see Variables).
   - The "Add to Doc Center" sequence adds the Pdf file to the student's records in the Document Center (see next step).
   - The "Drop Course" sequence removes the course from the students course list (see below).



6. The "Add to Doc Center" sequence creates and saves the Pdf file of the student's course drop request.

   - CreateDocument takes the **Pdf** variable as in-argument, returns the **StuDrop** (DocumentEntity) variable, and uses the **studentid** variable (see Variables).

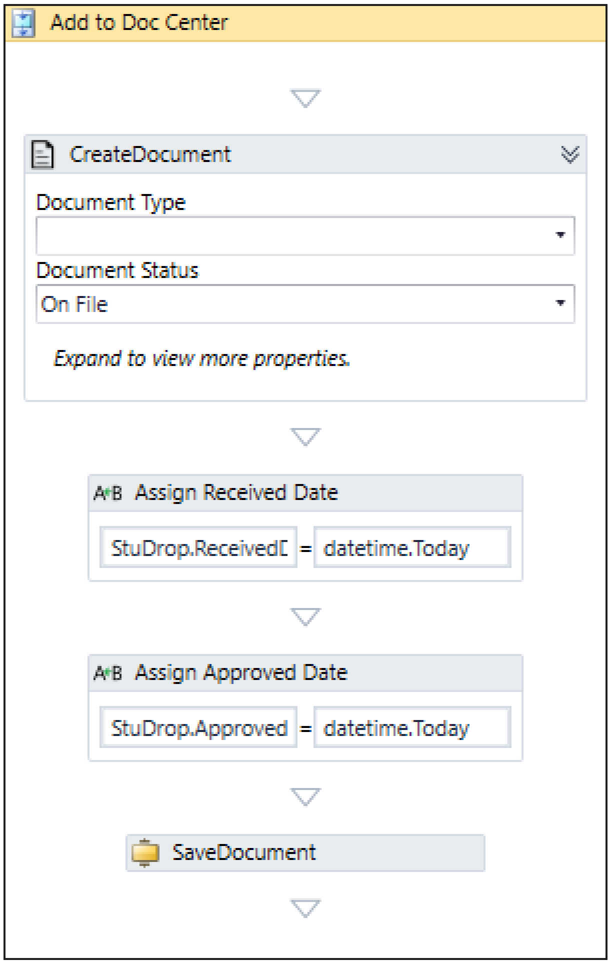
     It also specifies values for the following properties:

     - **Date Requested** (datetime.Today)
     - **Due Date** (datetime.Today)
     - **Document Type** (use drop-down to select, e.g., 148)
     - **Image FileName** ("StudentDropCourse.pdf")
     - **Notes** (e.g., "Student Drop Course Form Submitted Online")
     - **Status** (use drop-down to select, e.g., "On File")

   - Assign Received Date assigns the value "datetime.Today" to **StuDrop.ReceivedDate**.

   - Assign Approved Date assigns the value "datetime.Today" to the **StuDrop.ApprovedDate**.

   - SaveDocument saves the **StuDrop** (DocumentEntity).



Document Type

Document Status

On File

Expand to view more properties.

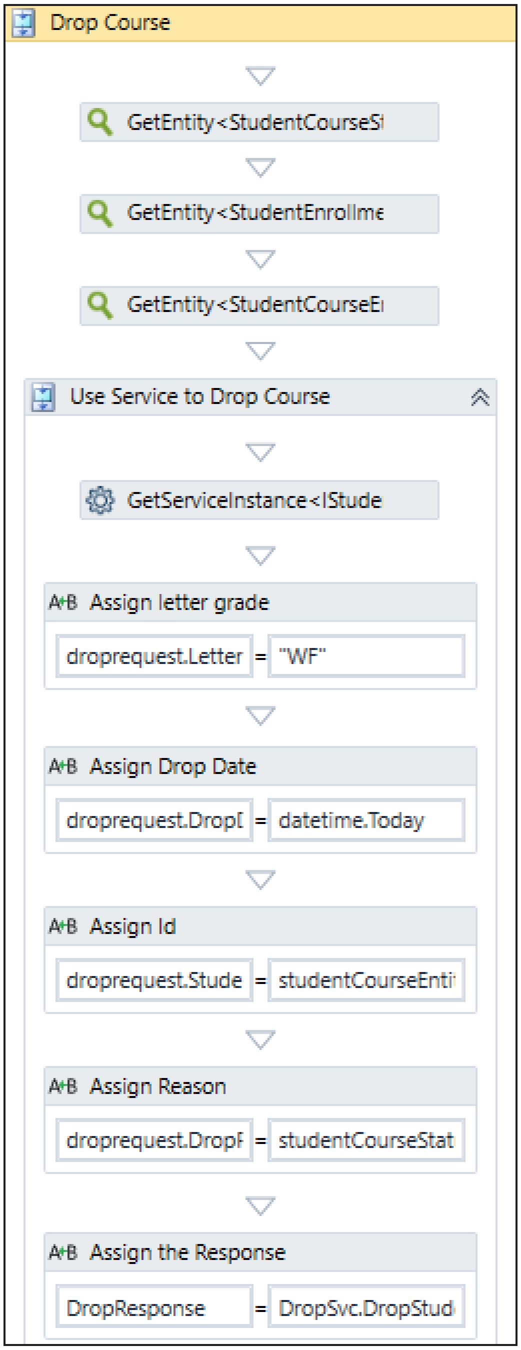
StuDrop.ReceivedI = datetime.Today

StuDrop.Approved = datetime.Today

7. The "Drop Course" sequence removes the course from the student's course list and assigns required properties.

   - GetEntity<StudentCourseStatusChangeReasonEntity> takes the studentCourseStatusChangeReasonEntity.Id value and returns **studentCourseStatusChangeReasonEntity**.
   - GetEntity<StudentEnrollmentPeriodEntity> takes the studentEnrollmentPeriodEntity.Id value and returns **studentEnrollmentPeriodEntity**.
   - GetEntity<StudentCourseEntity> takes the studentCourseEntity.CourseId value and returns **studentCourseEntity**.
   - GetServiceInstance<IStudentCourseService> returns the **DropSvc** variable.
   - Assign activities set the following values:

To	Value
DropRequest.LetterGrade	"WF"
DropRequest.DropDate	datetime.Today
DropRequest.StudentEnrollmentScheduleId	studentCourseEntity.Id
DropRequest.DropReasonId	studentCourseStatusChangeReasonEntity.Id
DropResponse	DropSvc.DropStudentCourse(DropRequest)


Drop Course
GetEntity<StudentCourseSt
GetEntity<StudentEnrollme
GetEntity<StudentCourseEı
Use Service to Drop Course
GetServiceInstance<IStude
A+B Assign letter grade
droprequest.Letter = "WF"
A+B Assign Drop Date
droprequest.DropI = datetime.Today
A+B Assign Id
droprequest.Stude = studentCourseEnti
A+B Assign Reason
droprequest.DropF = studentCourseStat
A+B Assign the Response
DropResponse = DropSvc.DropStud

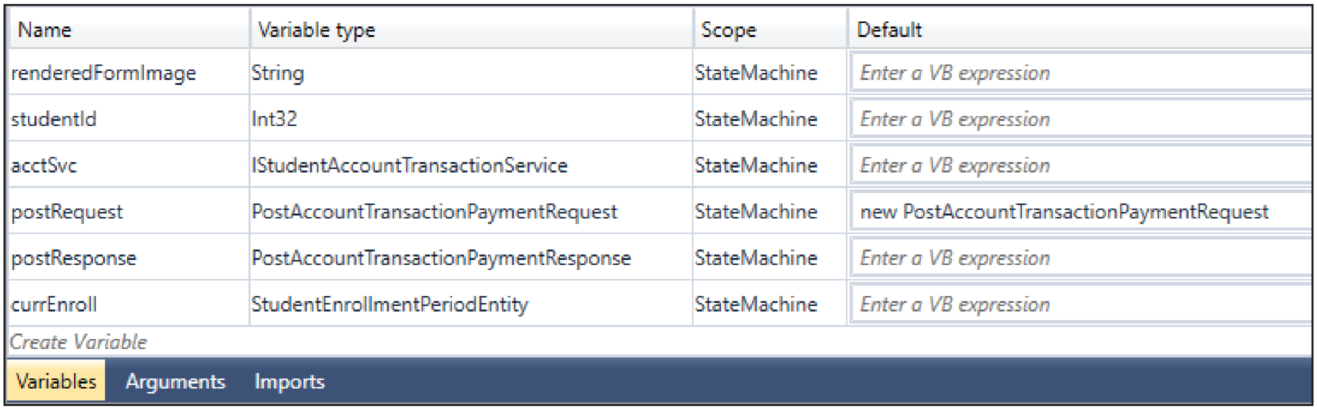
Note: Insert LogLine and LogObject activities at various points in the workflow as needed.

IStudentAccountTransactionService - Post Account Transaction Payment

The GetServiceInstance<> activity can be used to invoke the IStudentAccountTransactionService method. One of the operations that can be executed using the IStudentAccountTransactionService method is the PostAccountTransactionPayment operation. The response will indicate whether a student payment was received.

Workflow Example

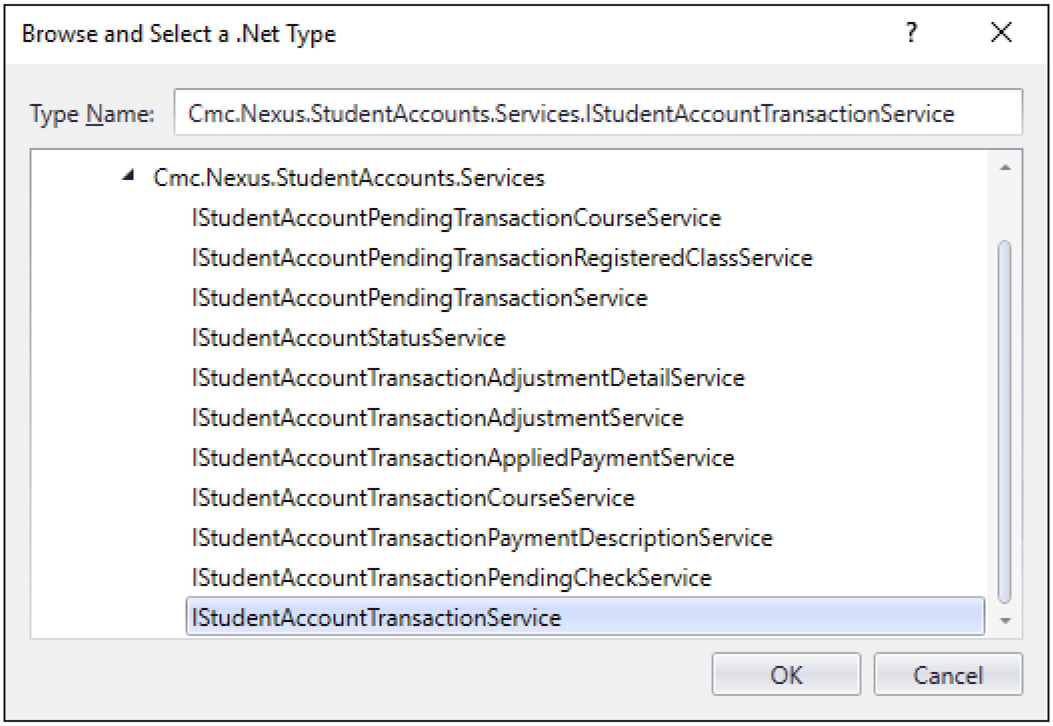
1. In Workflow Composer, create the following **Variables**. Be sure to use the indicated variable types and defaults.



Name	Variable type	Scope	Default
renderedFormImage	String	StateMachine	*Enter a VB expression*
studentId	Int32	StateMachine	*Enter a VB expression*
acctSvc	IStudentAccountTransactionService	StateMachine	*Enter a VB expression*
postRequest	PostAccountTransactionPaymentRequest	StateMachine	new PostAccountTransactionPaymentRequest
postResponse	PostAccountTransactionPaymentResponse	StateMachine	*Enter a VB expression*
currEnroll	StudentEnrollmentPeriodEntity	StateMachine	*Enter a VB expression*
Create Variable			

2. Drag the **GetServiceInstance** activity into the workflow, click ▾, and select **Browse for Type**.

   Select **IStudentAccountTransactionService** and click **OK**.

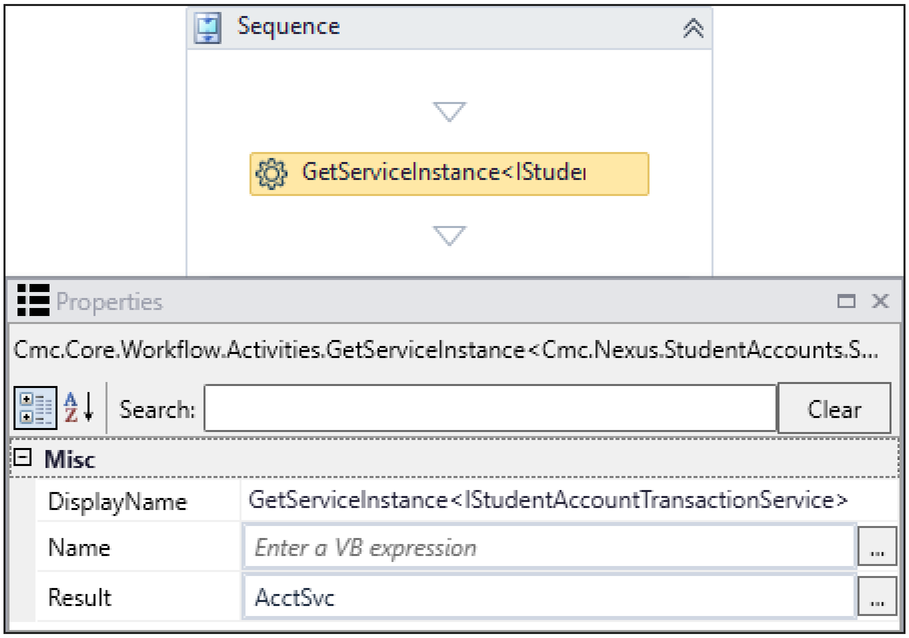


Browse and Select a .Net Type

Type Name: Cmc.Nexus.StudentAccounts.Services.IStudentAccountTransactionService

- Cmc.Nexus.StudentAccounts.Services
  - IStudentAccountPendingTransactionCourseService
  - IStudentAccountPendingTransactionRegisteredClassService
  - IStudentAccountPendingTransactionService
  - IStudentAccountStatusService
  - IStudentAccountTransactionAdjustmentDetailService
  - IStudentAccountTransactionAdjustmentService
  - IStudentAccountTransactionAppliedPaymentService
  - IStudentAccountTransactionCourseService
  - IStudentAccountTransactionPaymentDescriptionService
  - IStudentAccountTransactionPendingCheckService
  - IStudentAccountTransactionService

3. In the **Result** field of the GetServiceInstance property window, specify the variable of type IStudentAccountTransactionService created above.



Cmc.Core.Workflow.Activities.GetServiceInstance<Cmc.Nexus.StudentAccounts.S...

DisplayName	GetServiceInstance<IStudentAccountTransactionService>
Name	Enter a VB expression
Result	AcctSvc

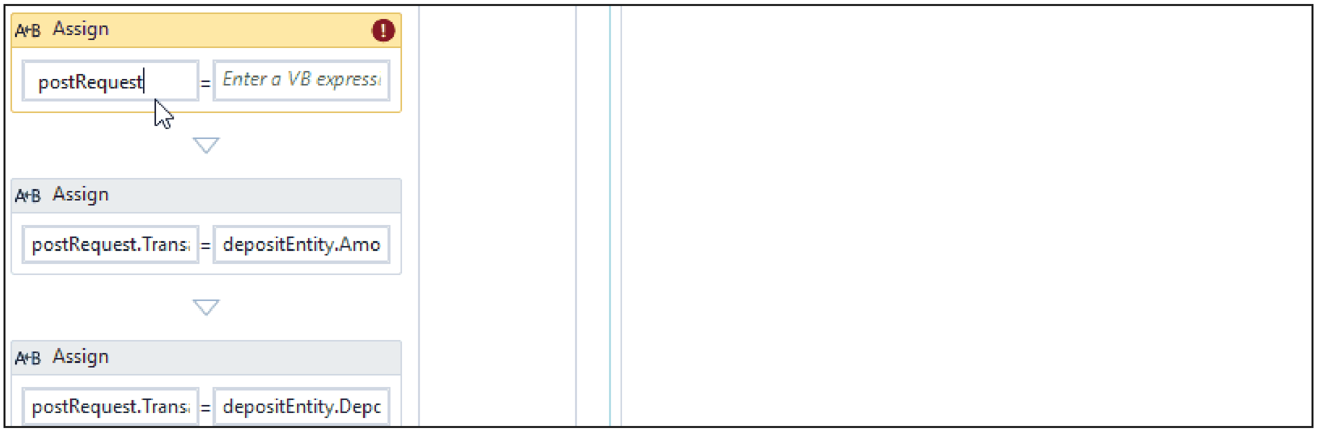
4. Drag **Assign** activities into the workflow for the following functions of the postRequest operation.

Assignments for Student Payment Transactions

Assign Activity		Notes
To	Value	
postRequest.StudentId	studentId	Variable created in your workflow or hard-coded value (see Notes)
postRequest.TransactionAmount	depositEntity.Amount	
postRequest.TransactionDate	depositEntity.DepositReceivedDate	
postRequest.PaymentType	Specify a Payment Type code (enclosed in quotation marks). The system-defined codes in Anthology Student are: • "C" for Cash • "E" for EFT • "H" for Check • "N" for Non-Cash • "R" for Credit Card	
postRequest.PaymentMode	PaymentMode.Normal	
postRequest.StudentEnrollmentPeriodId	currEnroll.Id Insert a LookupCurrentEnrollmentPeriod (V2) activity above the Assign statement for postRequest.StudentEnrollmentPeriodId to retrieve the current enrollment period associated with the studentId.	
If Cash Drawer Sessions are used, three additional assignments are required:		
CashDrawerId		
CashDrawerSessionId		
CashierId		

Notes:

When the GetServiceInstance activity is inserted into the workflow, you can use Intellisense on the variables in the Assign statements to see the available options. Type the request followed by a period to trigger Intellisense. Use the down arrow key to scroll through the available values and press Enter to select a value. The tooltip shows the variables and valid data types.

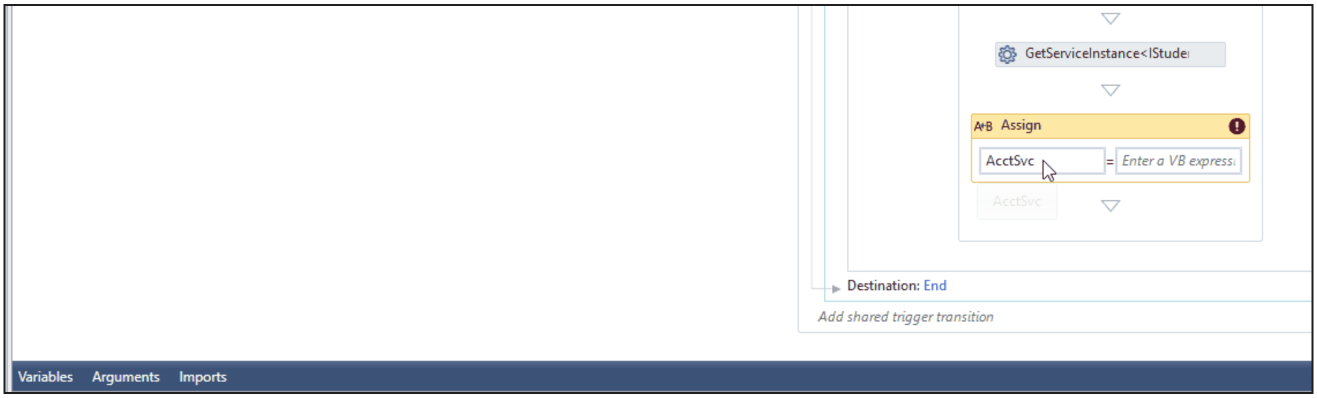


postRequest = Enter a VB express

postRequest.Trans = depositEntity.Amo

postRequest.Trans = depositEntity.Depc

Once the main method has been selected, Intellisense can then be used to see how to call (typically with a request and returning a response) the variable. Note the tooltip.



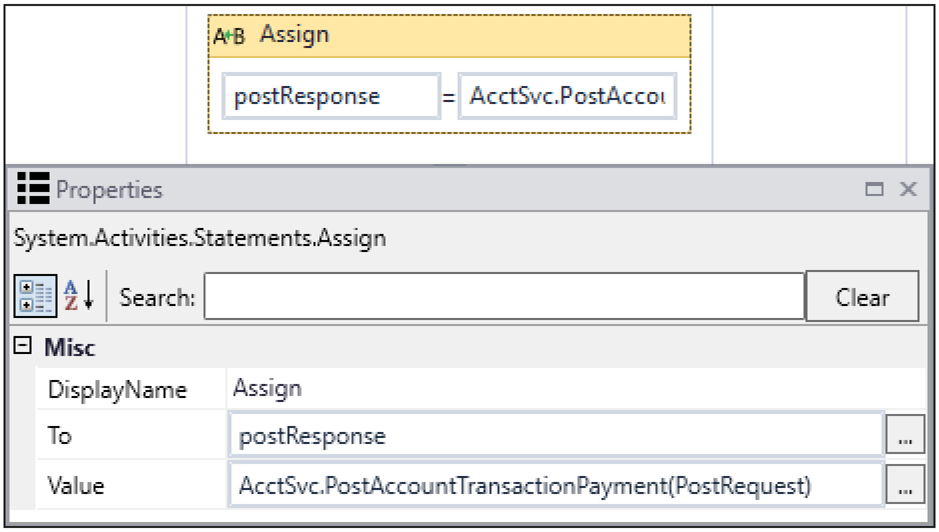
AcctSvc = Enter a VB express

Destination: End

Add shared trigger transition

5. Drag an **Assign** activity into the workflow to assign the response to the account transaction request.

   - In the "To" field specify: **postResponse** (This is a variable created in above.)
   - In the "Value" field specify: **AcctSvc.PostAccountTransactionPayment(PostRequest)** (Where "postRequest" is a variable created above.)



postResponse = AcctSvc.PostAccou

System.Activities.Statements.Assign

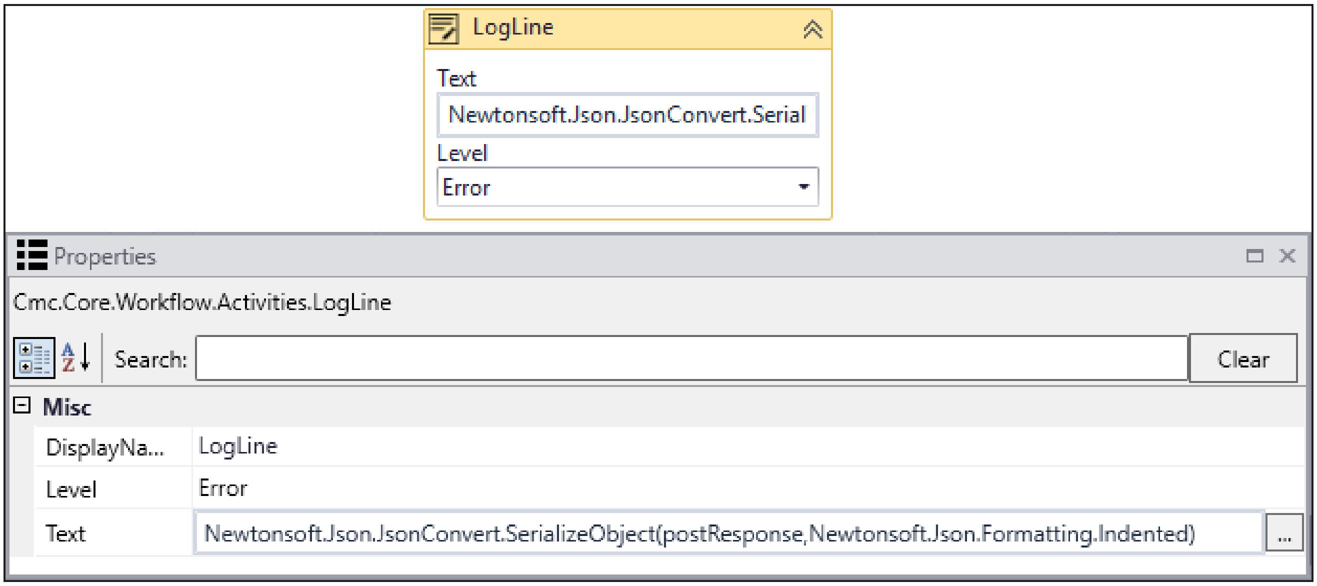
Misc

DisplayName	Assign
To	postResponse
Value	AcctSvc.PostAccountTransactionPayment(PostRequest)

6. To capture the result of the service response, insert WriteLine, LogLine, or any other activities as appropriate for your workflow.

   For example, you can use a LogLine activity with the following properties to capture the response after the call to the Student Account Transaction Service.

   - Level: **Error**
   - Text: **Newtonsoft.Json.JsonConvert.SerializeObject(postResponse,Newtonsoft.Json.Formatting.Indented)**



LogLine

Text: Newtonsoft.Json.JsonConvert.Serial

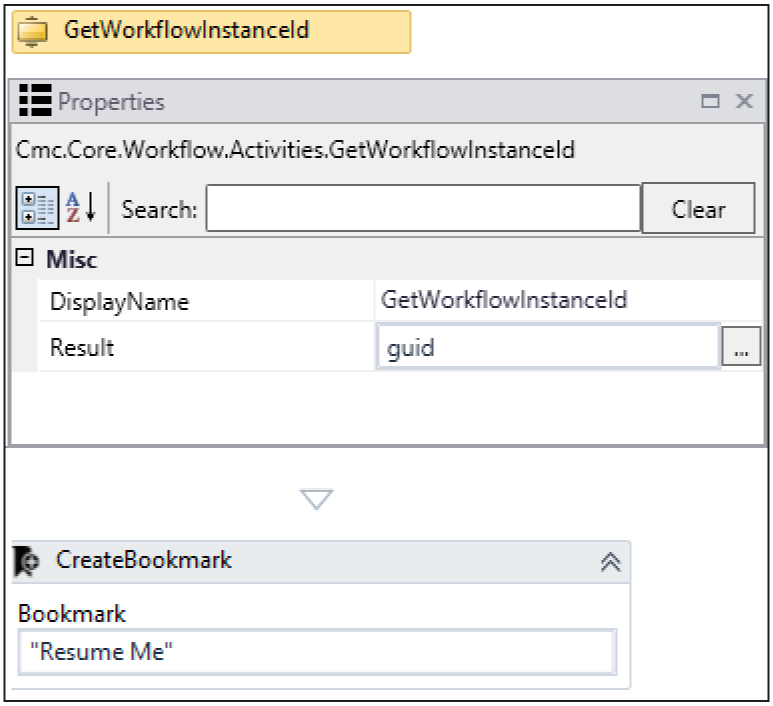
Level: Error

Cmc.Core.Workflow.Activities.LogLine

DisplayNa...	LogLine
Level	Error
Text	Newtonsoft.Json.JsonConvert.SerializeObject(postResponse,Newtonsoft.Json.Formatting.Indented)

GetWorkflowInstanceId

The GetWorkflowInstanceId activity retrieves the workflow instance id of the currently executing workflow. This activity is used within long running workflows prior to the CreateBookmark activity. The Id returned from this activity needs to be passed into the ResumeBookmark activity.



Cmc.Core.Workflow.Activities.GetWorkflowInstanceId

Search:

Misc

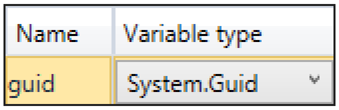
DisplayName	GetWorkflowInstanceId
Result	guid

CreateBookmark

Bookmark: "Resume Me"

Properties

GetWorkflowInstanceId Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Result	`OutArgument<Guid>`	Yes	The OutArgument holds the workflow instance Id associated with this workflow. The variable type for the OutArgument is System.Guid.  Name: guid Variable type: System.Guid

To see how GetWorkflowInstanceId can be used in a workflow, refer to:

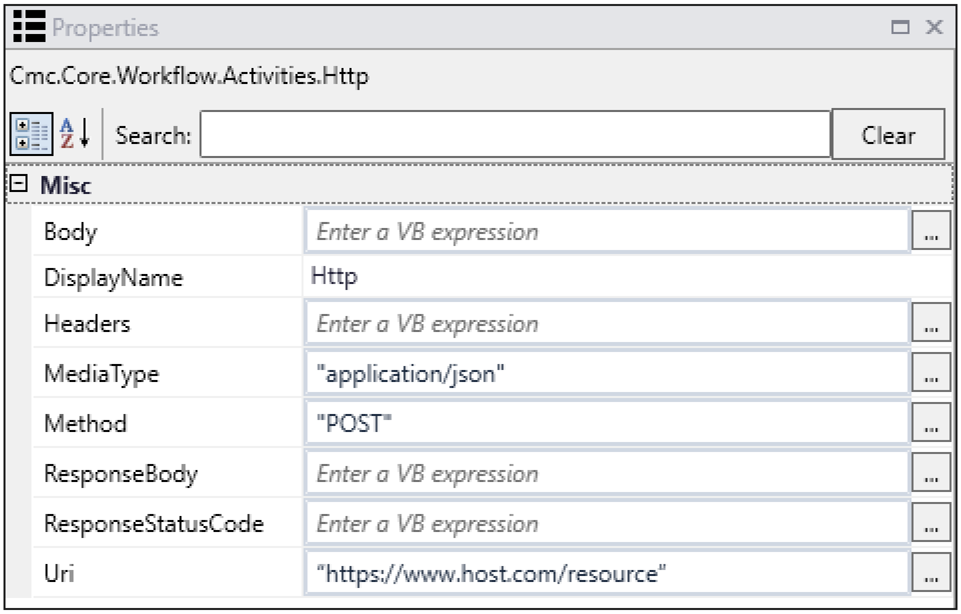
- Create a Long Running Workflow

Http

You can use the Http activity to integrate the Anthology platform with external systems. The activity supports REST and SOAP web services. It enables posting messages, retrieving data, returning status results, and other actions related to a specific resource.

Anthology applications use this activity to post messages to the Azure Service Bus and Azure Logic Apps, Microsoft Flow and Office 365, as well as any other external Web APIs. Anthology Student Finance, HR & Payroll uses this activity to integrate Anthology Student and Microsoft Dynamics 365.

The Http activity will execute (send) a request and you will get a response from the Url end-point that is being posted to. For the SendToAzureServiceBus activity, the workflow logic cannot depend on getting an immediate result from the process –- all you will know is that the message was successfully queued. If you want to get or post data and want to know the result immediately (synchronously), use the Http activity. For more information, see example Http vs. SendToAzureServiceBus.



Cmc.Core.Workflow.Activities.Http

Search:

Misc

Body	Enter a VB expression
DisplayName	Http
Headers	Enter a VB expression
MediaType	"application/json"
Method	"POST"
ResponseBody	Enter a VB expression
ResponseStatusCode	Enter a VB expression
Uri	"https://www.host.com/resource"

Properties

Http Activity Properties

Property	Value	Required	Notes
Body	`InArgument<String>`	No	Represents data to be transferred in the HTTP request to the server.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Headers	`InArgument<StringDictionary>`	No	Represents the name/value pairs that are transferred in the request.

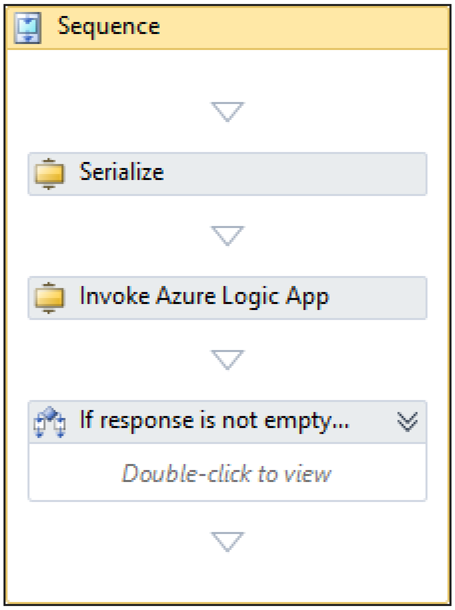
Property	Value	Required	Notes
MediaType	`InArgument<String>`	Yes	The media type of the body of the request. (e.g., "application/json"). Media type is typically used with POST, PUT, PATCH methods/verbs.
Method	`InArgument<String>`	Yes	HTTP method that indicates the action to be performed for a given resource: GET, POST, PUT, HEAD, DELETE, PATCH, CONNECT, OPTIONS, TRACE
ResponseBody	`OutArgument<String>`	No	The response body returned from the server.
ResponseStatusCode	`OutArgument<HttpStatusCode>`	No	Represents the HTTP response status code issued by the server in response to the request (e.g., 200, 401, 500, etc.).
URI	`InArgument<String>`	Yes	The Universal Resource Identifier to which the request will be made (e.g., "https://www.host.com/resource").

Examples

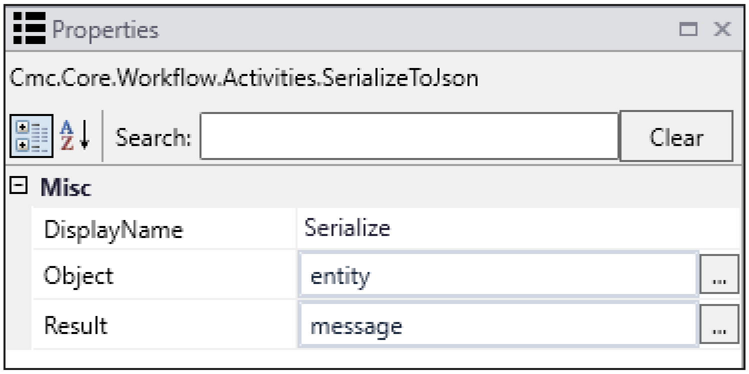
Invoke an Azure Logic App

The workflow example below is available on GitHub. Refer to the instructions at https://github.com/campusmanagement/workflow-samples/blob/master/README.md.

This example shows how the Http activity can be used to invoke an Azure logic app. The xaml file is available here: Cmc.Nexus.Crm.Entities.TaskEntity_SavingEvent_Sample - Azure Logic Apps.xaml.



1. The SerializeToJson activity serializes an input argument object named "entity" and produces the output string named "message".



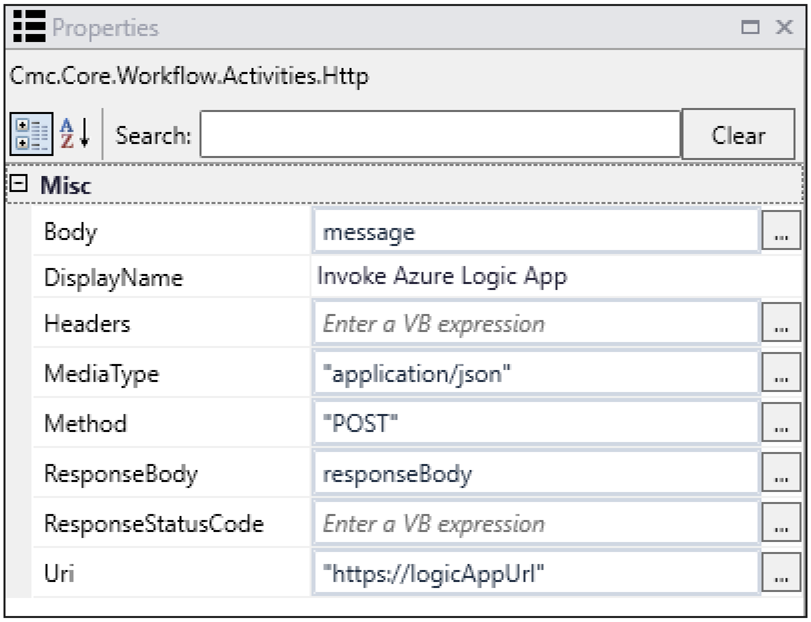
Cmc.Core.Workflow.Activities.SerializeToJson

Misc

DisplayName	Serialize
Object	entity
Result	message

2. The next activity is an Http activity. It:

   - Uses the serialized "message" string as input argument in the Body property.
   - Defines the input as MediaType = "application/json".
   - Invokes the "POST" method.
   - Creates the output argument named "responseBody".
   - Sends the output to URI = "https://logicAppUrl" using the POST method.



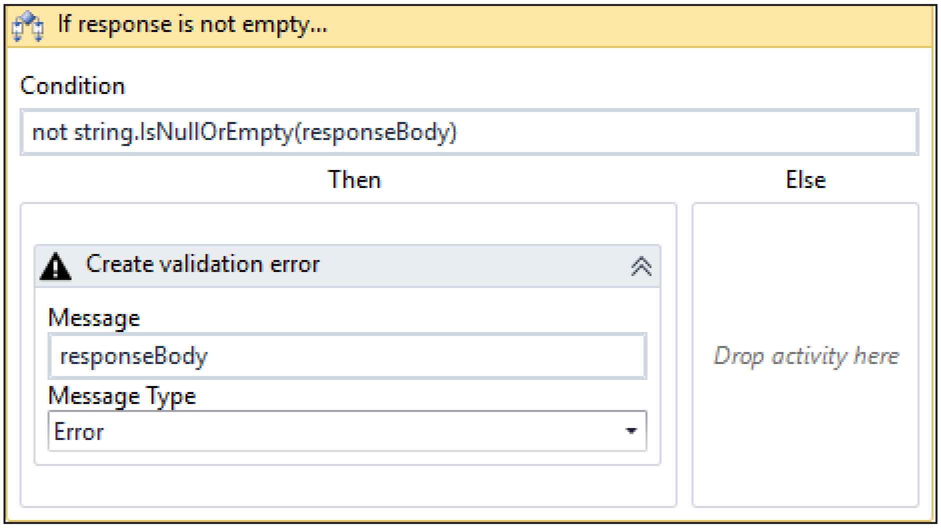
Cmc.Core.Workflow.Activities.Http

Body	message
DisplayName	Invoke Azure Logic App
Headers	*Enter a VB expression*
MediaType	"application/json"
Method	"POST"
ResponseBody	responseBody
ResponseStatusCode	*Enter a VB expression*
Uri	"https://logicAppUrl"

3. The If activity validates the output from the Http activity using the following Boolean condition:

   not string.IsNullOrEmpty(responseBody)

   The string.IsNullOrEmpty(responseBody) method checks whether the specified string (i.e., responseBody) is null or an empty string ("").



If response is not empty...

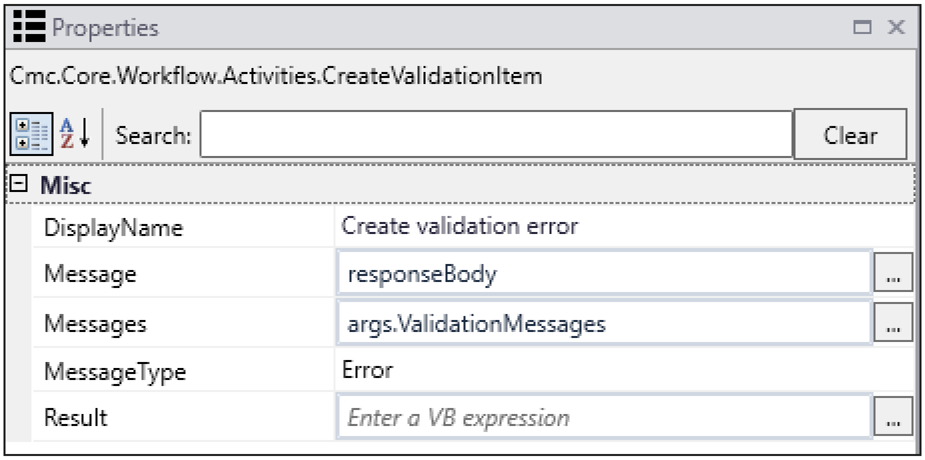
Condition: not string.IsNullOrEmpty(responseBody)

Create validation error

Message: responseBody

Message Type: Error

- If the condition is met (i.e., the responseBody string is empty), the CreateValidationItem activity creates an error message.



Properties

Cmc.Core.Workflow.Activities.CreateValidationItem

Search: Clear

Misc

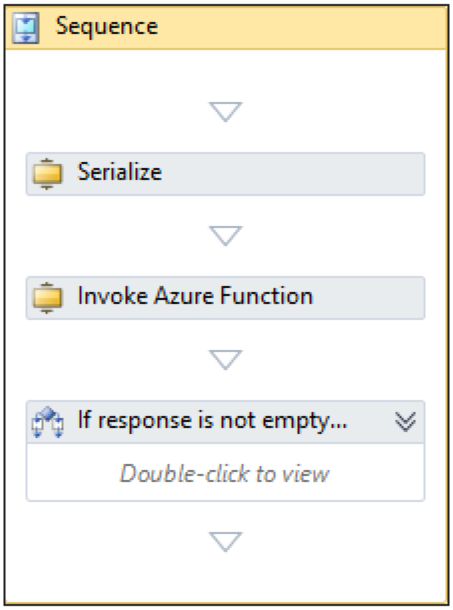
DisplayName	Create validation error
Message	responseBody
Messages	args.ValidationMessages
MessageType	Error
Result	*Enter a VB expression*

- If the condition is not met, the responseBody string is sent to the URI specified in the Http activity

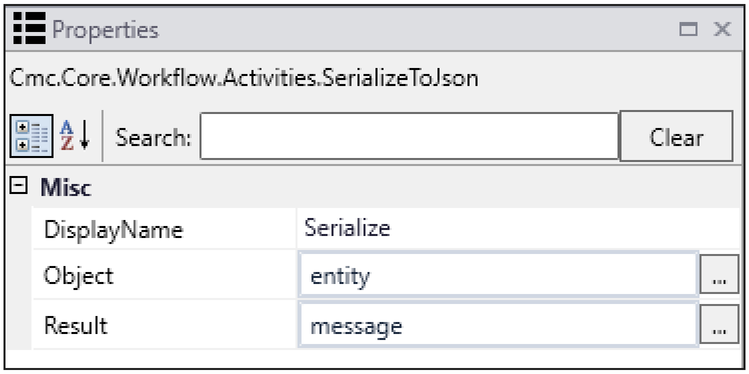
Invoke an Azure Function

The workflow example below is available on GitHub. Refer to the instructions at https://github.com/campusmanagement/workflow-samples/blob/master/README.md.

This example shows how the Http activity can be used to invoke an Azure function. The xaml file is available here: Cmc.Nexus.Crm.Entities.TaskEntity_SavingEvent_Sample - Azure Functions.xaml.



1. The SerializeToJson activity serializes an input argument object named "entity" and produces the output string named "message".

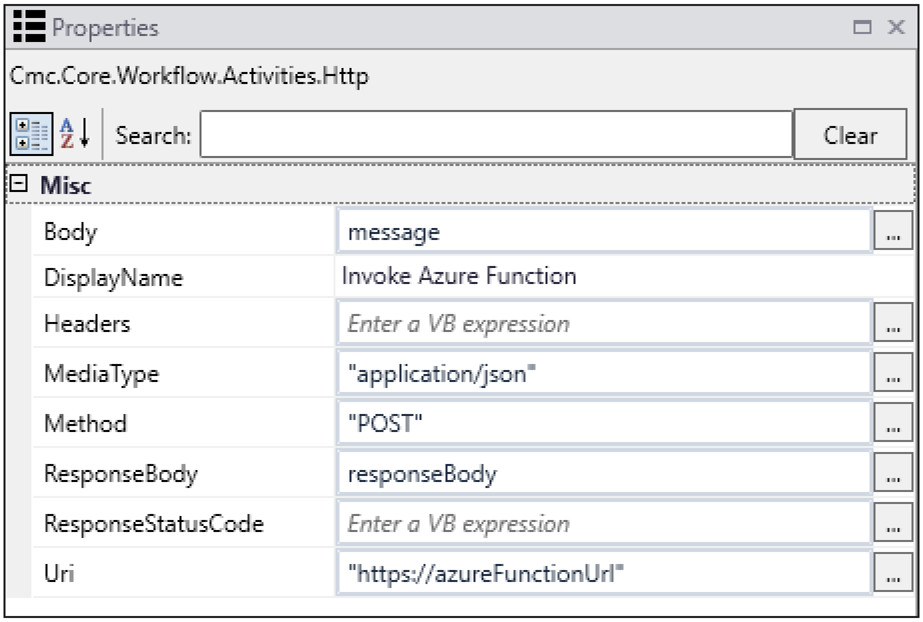


Cmc.Core.Workflow.Activities.SerializeToJson

DisplayName	Serialize
Object	entity
Result	message

2. The next activity is an Http activity. It:

   - Uses the serialized "message" string as input argument in the Body property.
   - Defines the input as MediaType = "application/json".
   - Invokes the "POST" method.
   - Creates the output argument named "responseBody".
   - Sends the output to URI = "https://azureFunctionUrl" using the POST method.



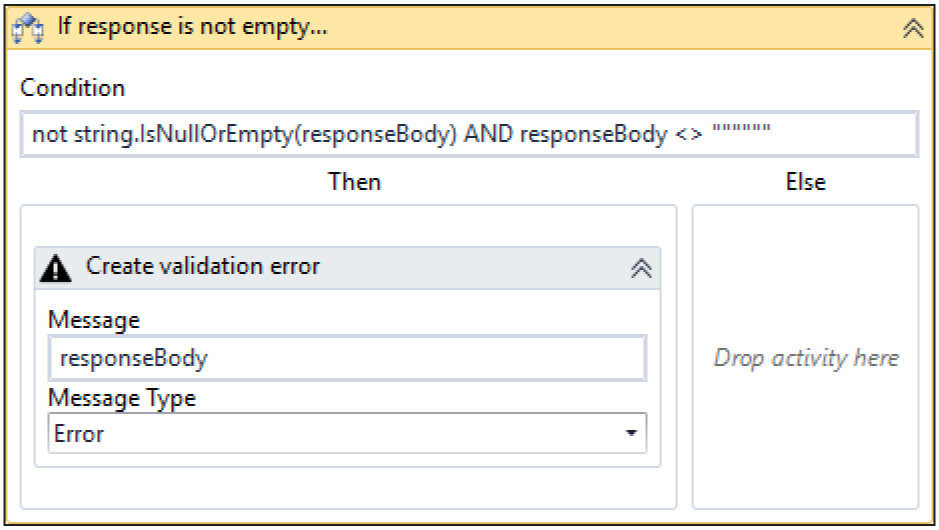
Cmc.Core.Workflow.Activities.Http

Body	message
DisplayName	Invoke Azure Function
Headers	*Enter a VB expression*
MediaType	"application/json"
Method	"POST"
ResponseBody	responseBody
ResponseStatusCode	*Enter a VB expression*
Uri	"https://azureFunctionUrl"

3. The If activity validates the output from the Http activity using the following Boolean condition:

   not string.IsNullOrEmpty(responseBody) AND responseBody <> """"""

   The string.IsNullOrEmpty(responseBody) method checks whether the specified string (i.e., responseBody) is null or an empty string ("").



If response is not empty...

Condition

not string.IsNullOrEmpty(responseBody) AND responseBody <> """"""

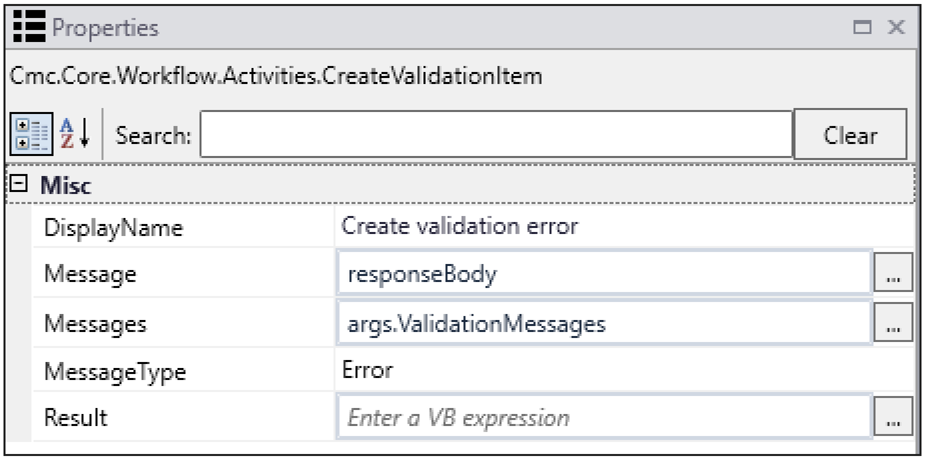
Message

responseBody

Message Type

Error

- If the condition is met (i.e., the responseBody string is empty), the CreateValidationItem activity creates an error message.



Cmc.Core.Workflow.Activities.CreateValidationItem

DisplayName	Create validation error
Message	responseBody
Messages	args.ValidationMessages
MessageType	Error
Result	Enter a VB expression

- If the condition is not met, the responseBody string is sent to the URI specified in the Http activity

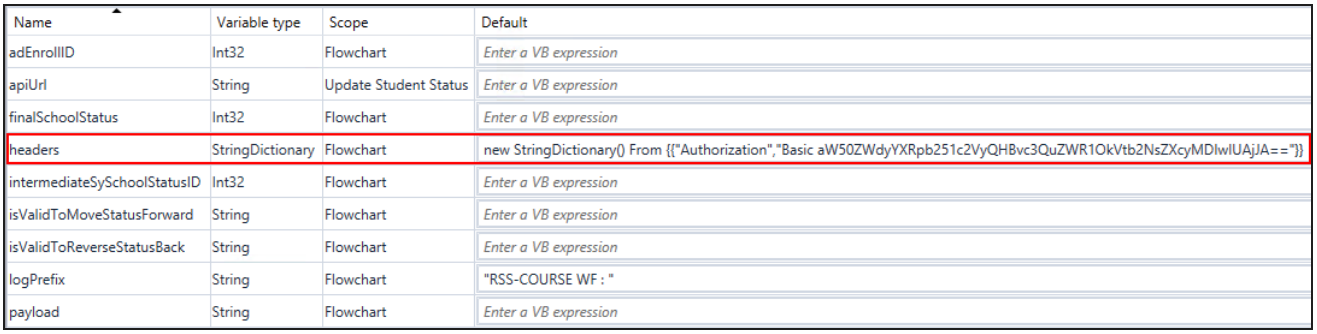
Use the Http Header for Authentication

The Headers field in the Http activity can be used to pass an authentication key for API calls. The value in the Headers field is based on the StringDictionary class.

a. Our first example uses basic authentication.

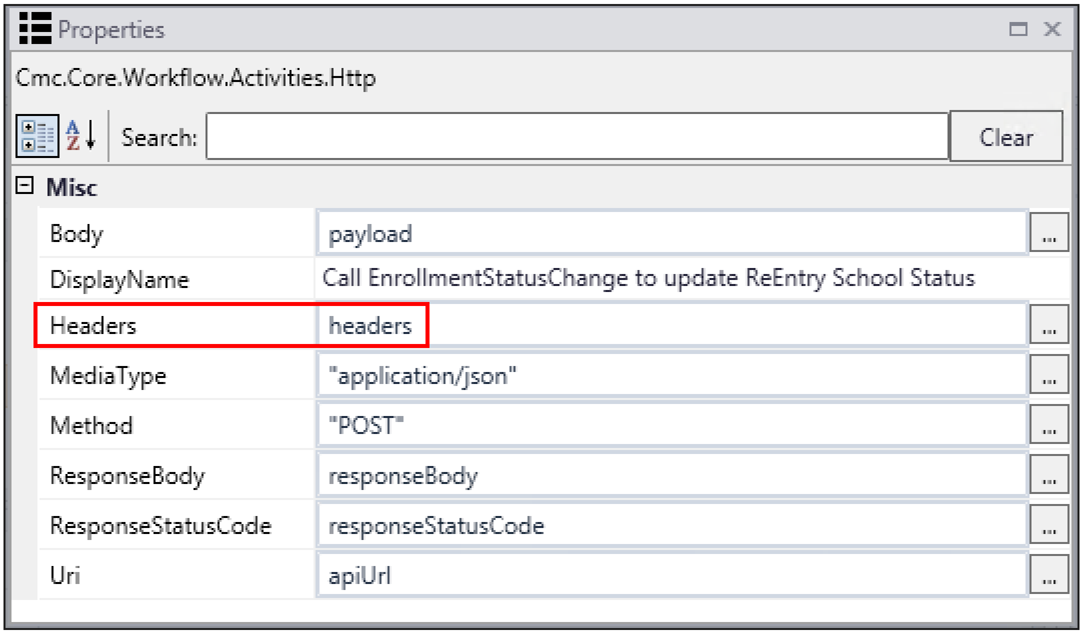
   A variable of Type StringDictionary is defined as follows:

   new StringDictionary() From {{"Authorization","Basic <Authentication Key>"}}



Name	Variable type	Scope	Default
adEnrollID	Int32	Flowchart	*Enter a VB expression*
apiUrl	String	Update Student Status	*Enter a VB expression*
finalSchoolStatus	Int32	Flowchart	*Enter a VB expression*
headers	StringDictionary	Flowchart	new StringDictionary() From {{"Authorization","Basic aW50ZWdyYXRpb251c2VyQHBvc3QuZWR1OkVtb2NsZXcyMDIwIUAjJA=="}}
intermediateSySchoolStatusID	Int32	Flowchart	*Enter a VB expression*
isValidToMoveStatusForward	String	Flowchart	*Enter a VB expression*
isValidToReverseStatusBack	String	Flowchart	*Enter a VB expression*
logPrefix	String	Flowchart	"RSS-COURSE WF : "
payload	String	Flowchart	*Enter a VB expression*

The variable is specified in the Headers field of the Http activity.

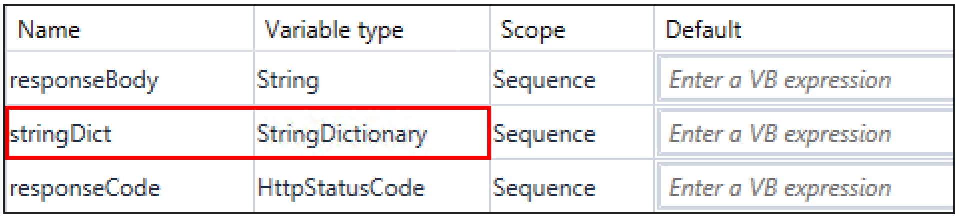


Cmc.Core.Workflow.Activities.Http

Body	payload
DisplayName	Call EnrollmentStatusChange to update ReEntry School Status
Headers	headers
MediaType	"application/json"
Method	"POST"
ResponseBody	responseBody
ResponseStatusCode	responseStatusCode
Uri	apiUrl

b. Our second example uses an ApiKey for authentication.

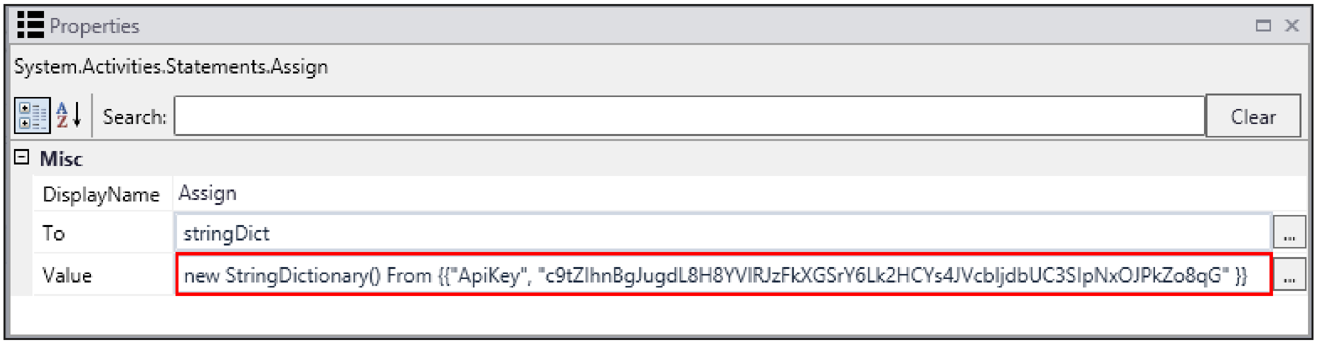
A StringDictionary variable is defined as follows:



Name	Variable type	Scope	Default
responseBody	String	Sequence	*Enter a VB expression*
stringDict	StringDictionary	Sequence	*Enter a VB expression*
responseCode	HttpStatusCode	Sequence	*Enter a VB expression*

An Assign activity is used to assign a value to the variable:

new StringDictionary() From {{"ApiKey", "<Authentication Key>" }}

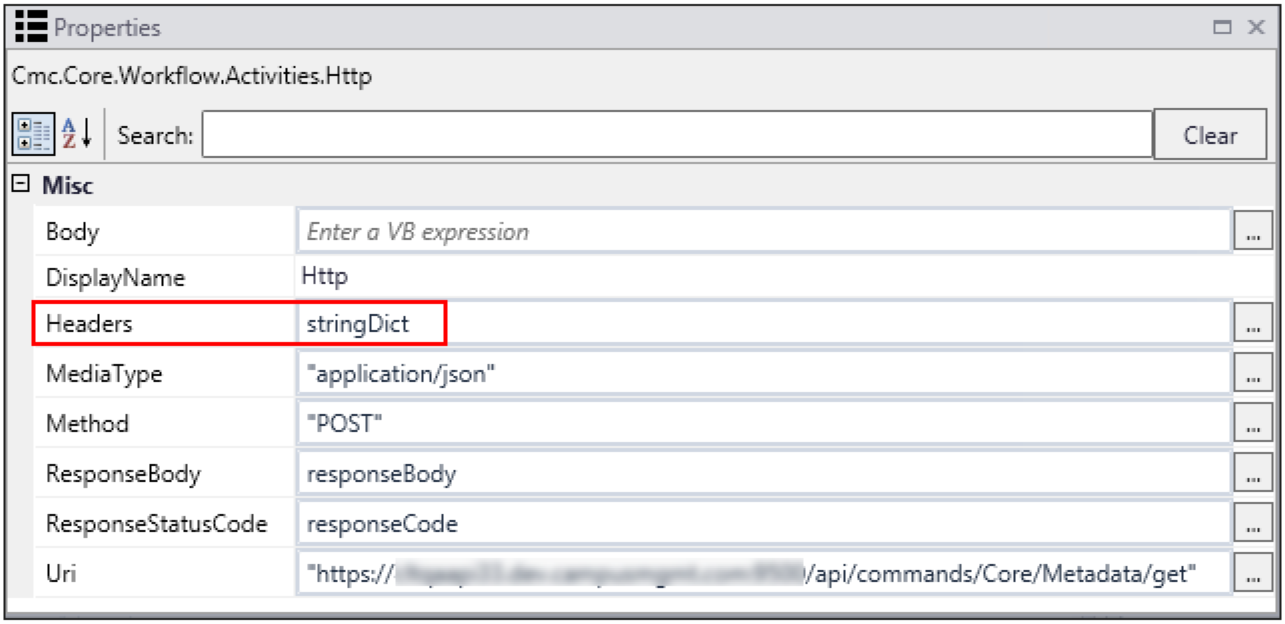


System.Activities.Statements.Assign

Misc

DisplayName	Assign
To	stringDict
Value	new StringDictionary() From {{"ApiKey", "c9tZIhnBgJugdL8H8YVIRJzFkXGSrY6Lk2HCYs4JVcbljdbUC3SIpNxOJPkZo8qG" }}

The variable is specified in the Headers field of the Http activity.



Cmc.Core.Workflow.Activities.Http

Misc

Body	*Enter a VB expression*
DisplayName	Http
Headers	stringDict
MediaType	"application/json"
Method	"POST"
ResponseBody	responseBody
ResponseStatusCode	responseCode
Uri	"https://[redacted]/api/commands/Core/Metadata/get"

Note: In VB.Net, the name/value pair of the API key is translated to lowercase before it is added to the string dictionary. For more information, see StringDictionary Class. This can cause authentication to fail. However, in Anthology Student 22.0 and later, if passing the ApiKey name/value pair in the header of the Http activity, any casing will be accepted.

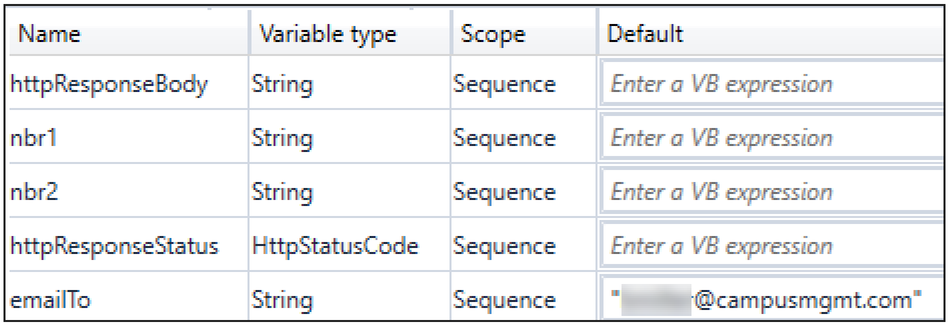
Http vs. SendToAzureServiceBus

To demonstrate the difference between the Http and SendToAzureServiceBus activities, we created a workflow that multiplies two numbers (24 x 365) and returns the result (8,760).

- The Http activity returns the result immediately to a workflow variable.
- The SendToAzureServiceBus activity sends the result to the service bus where it is processed by an application that is listening for messages. Then response is sent to the email address specified in the request.

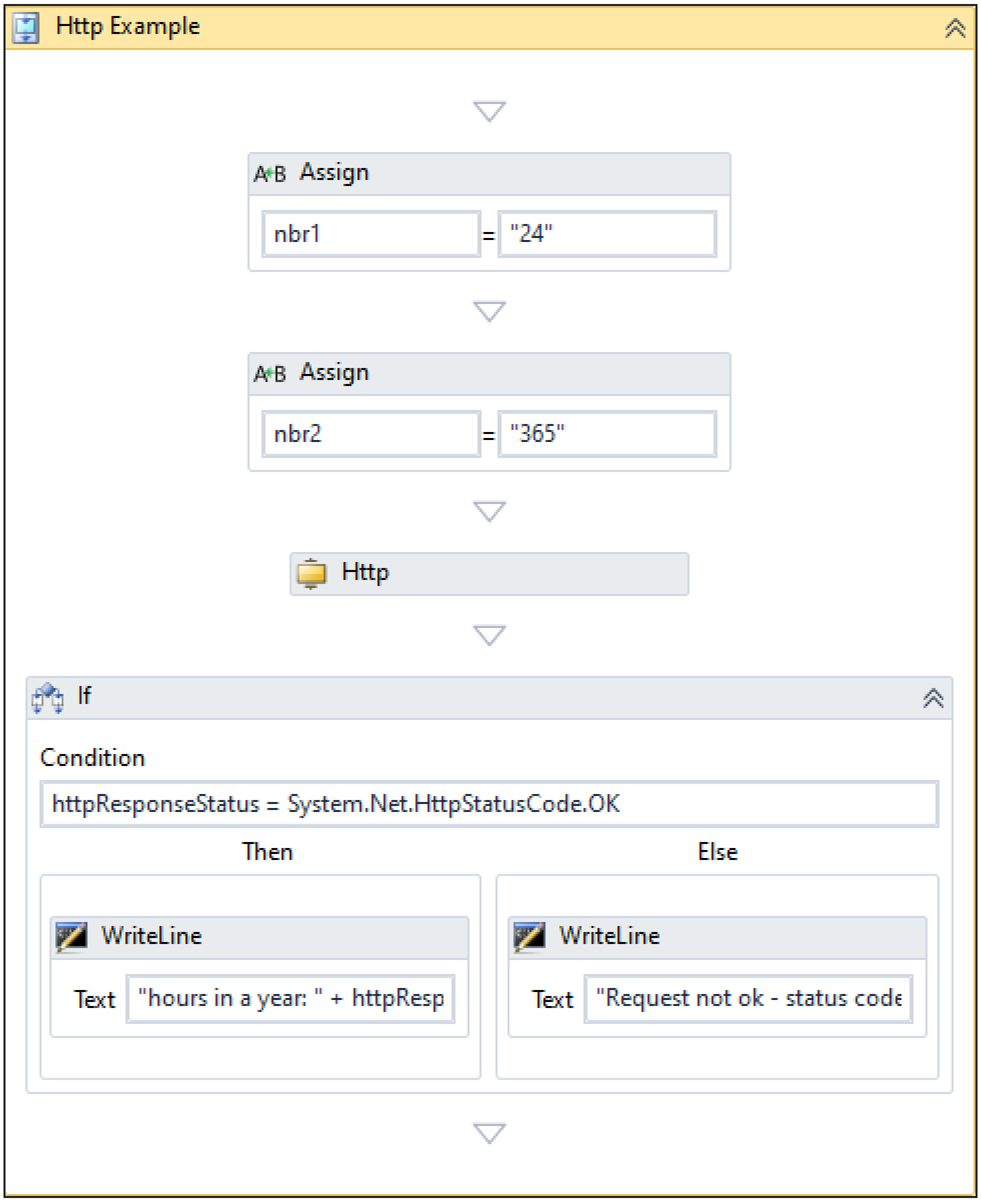
In more complex scenarios, the response from the service bus listener process could be a call back into another system –- or the service bus listener would forward the message to a 3rd party application to be posted to that system.

The workflow uses the following variables:



Name	Variable type	Scope	Default
httpResponseBody	String	Sequence	*Enter a VB expression*
nbr1	String	Sequence	*Enter a VB expression*
nbr2	String	Sequence	*Enter a VB expression*
httpResponseStatus	HttpStatusCode	Sequence	*Enter a VB expression*
emailTo	String	Sequence	"[redacted]@campusmgmt.com"

Http Activity



nbr1 = "24"

nbr2 = "365"

Condition
httpResponseStatus = System.Net.HttpStatusCode.OK

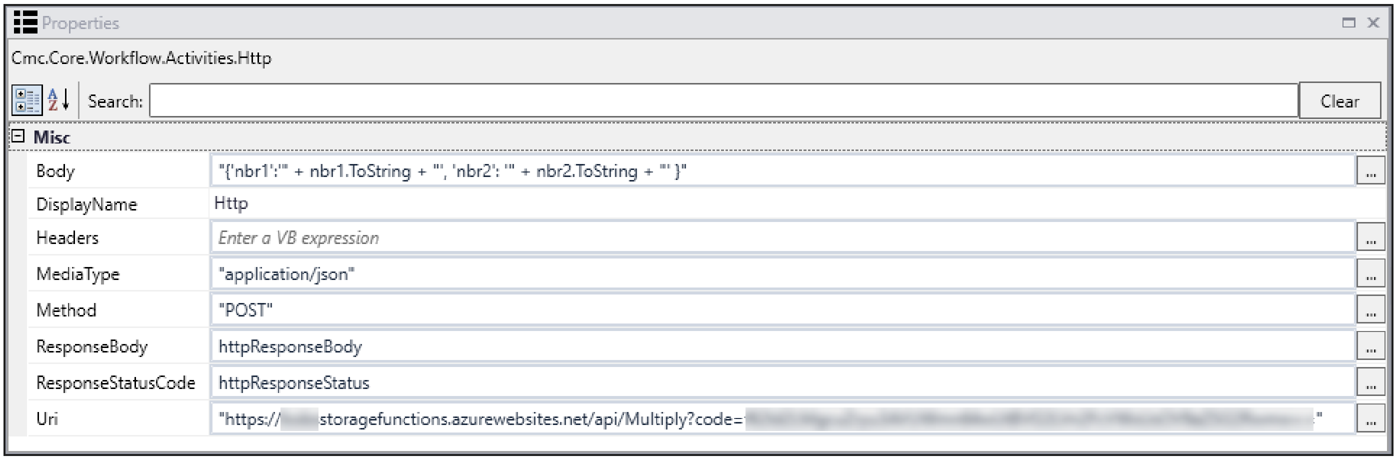
Text "hours in a year: " + httpResp

Text "Request not ok - status code

The Http activity:

- Uses the string assignments (nbr1 and nbr2) as input arguments in the Body property.
- Defines the input as MediaType = "application/json".
- Invokes the "POST" method.
- Creates the output argument named "httpResponseBody".
- Creates the output argument named "httpResponseStatus" whose value is checked in the If Condition.
- Sends the output to a Uri on an Azure web site that hosts an API.

The API multiplies the numbers 2 numbers in the request Body (nbr1 and nbr2) and returns the result.

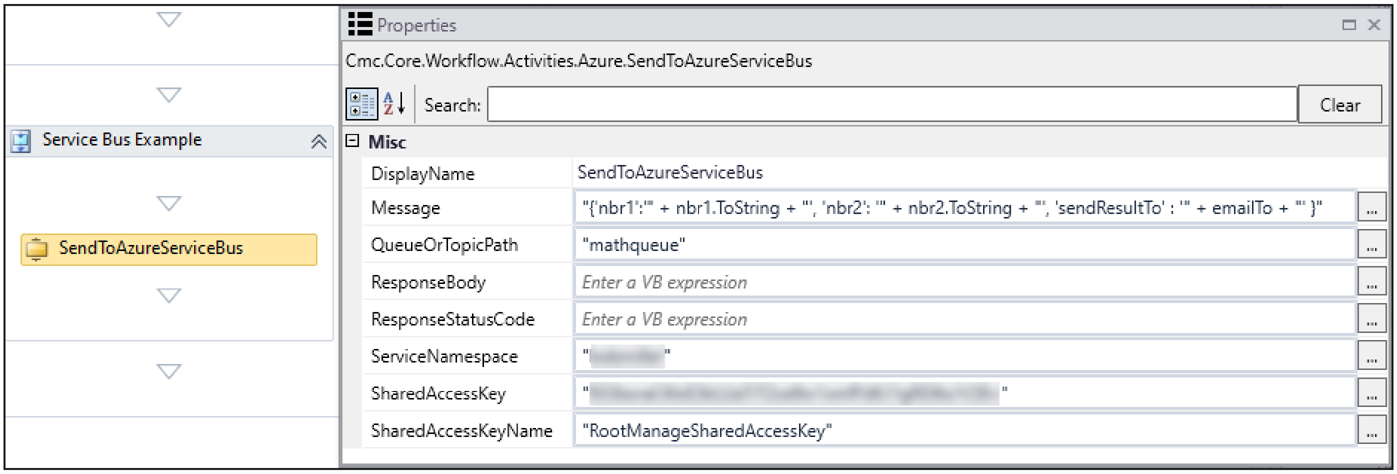


Cmc.Core.Workflow.Activities.Http

Misc

Body	"{'nbr1':'" + nbr1.ToString + "', 'nbr2': '" + nbr2.ToString + "' }"
DisplayName	Http
Headers	*Enter a VB expression*
MediaType	"application/json"
Method	"POST"
ResponseBody	httpResponseBody
ResponseStatusCode	httpResponseStatus
Uri	"https://[redacted]storagefunctions.azurewebsites.net/api/Multiply?code=[redacted]"

SendToAzureServiceBus Activity



Cmc.Core.Workflow.Activities.Azure.SendToAzureServiceBus

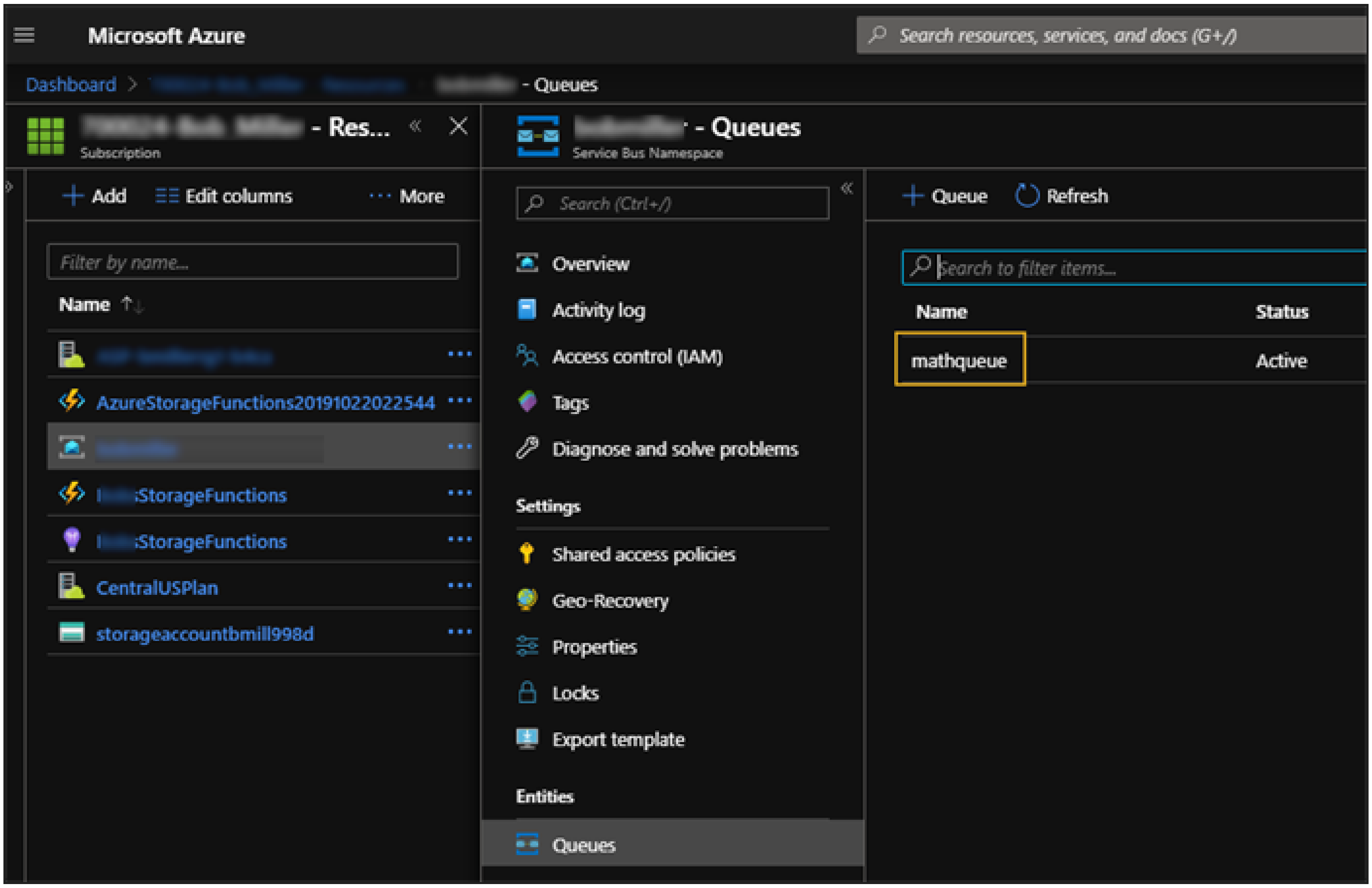
Misc

DisplayName	SendToAzureServiceBus
Message	"{'nbr1':'" + nbr1.ToString + "', 'nbr2': '" + nbr2.ToString + "', 'sendResultTo' : '" + emailTo + "' }"
QueueOrTopicPath	"mathqueue"
ResponseBody	*Enter a VB expression*
ResponseStatusCode	*Enter a VB expression*
ServiceNamespace	"[redacted]"
SharedAccessKey	"[redacted]"
SharedAccessKeyName	"RootManageSharedAccessKey"

The SendToAzureServiceBus activity:

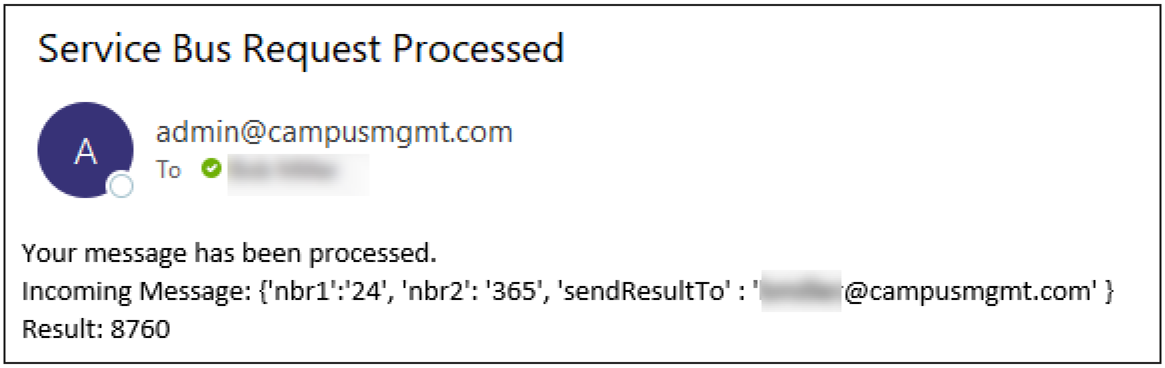
- Sends the string assignments (nbr1 and nbr2) and "emailTo" variable to the Azure Service Bus.
- Specifies the path for the Azure Service Bus as "mathqueue".
- Specifies the user's service name space and access key in Azure.

In Azure, the message is placed in the "mathqueue" and processed.



Name	Status
mathqueue	Active

When the service bus request is processed, an email is sent to the user.



Service Bus Request Processed

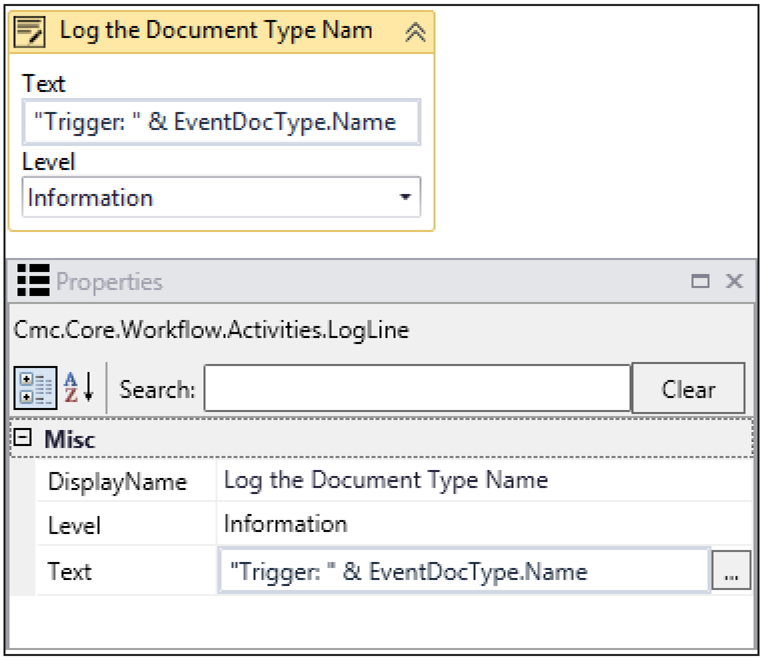
admin@campusmgmt.com

To [redacted]

Your message has been processed.
Incoming Message: {'nbr1':'24', 'nbr2': '365', 'sendResultTo' : '[redacted]@campusmgmt.com' }
Result: 8760

LogLine

The LogLine activity uses the Anthology logging infrastructure as opposed to the WriteLine (see Primitives), which only writes to the Windows console. LogLine is useful for processes such as IIS, Anthology Student, and Windows services that are not executing in console mode.



Text: "Trigger: " & EventDocType.Name

Level: Information

Cmc.Core.Workflow.Activities.LogLine

DisplayName	Log the Document Type Name
Level	Information
Text	"Trigger: " & EventDocType.Name

Properties

LogLine Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Level	`LoggerLevel`	Yes	Select a trace level from the drop-down list. The options are: • Trace • Debug • Information • Warning • Error • Fatal
Text	`InArgument<String>`	Yes	Input text string to include in the log file.

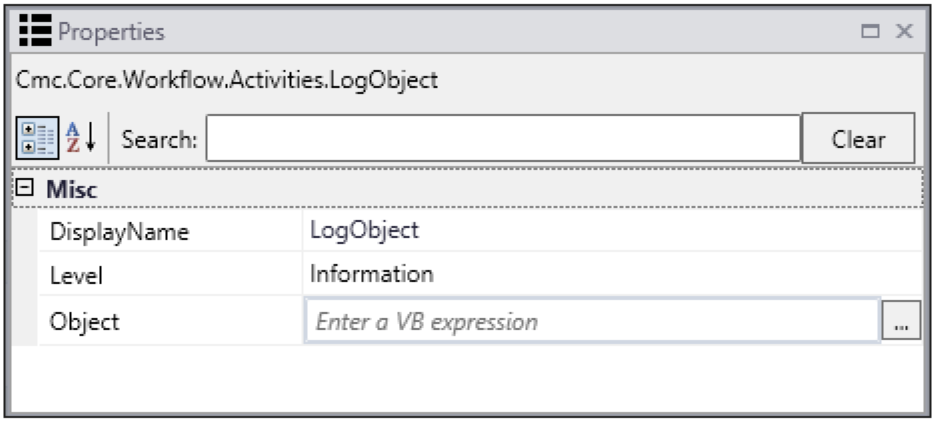
To see how LogLine can be used in a workflow, refer to:

- Check Approved Grants for Comments
- Create a Long Running Workflow

For information about configuring logging, refer to NLog.

LogObject

The LogObject activity initializes a new instance of the LogLine class. Use this activity to log everything being created on an entity.



Cmc.Core.Workflow.Activities.LogObject

DisplayName LogObject

Level Information

Object *Enter a VB expression*

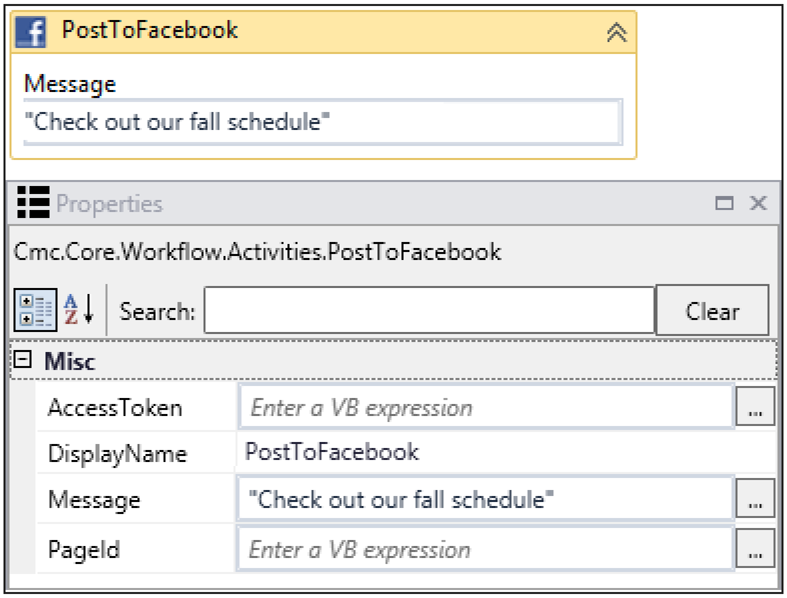
Properties

LogObject Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Level	`LoggerLevel`	Yes	Select a trace level from the drop-down list. The options are: • Trace • Debug • Information • Warning • Error • Fatal
Object	`InArgument<Object>`	Yes	Specify the name of an object in the Anthology data model, e.g., studentEntity.

PostToFacebook

The PostToFacebook activity enables you to display information on a Facebook page.



PostToFacebook
Message
"Check out our fall schedule"
Properties
Cmc.Core.Workflow.Activities.PostToFacebook
Search:
Clear
Misc

AccessToken	Enter a VB expression
DisplayName	PostToFacebook
Message	"Check out our fall schedule"
PageId	Enter a VB expression

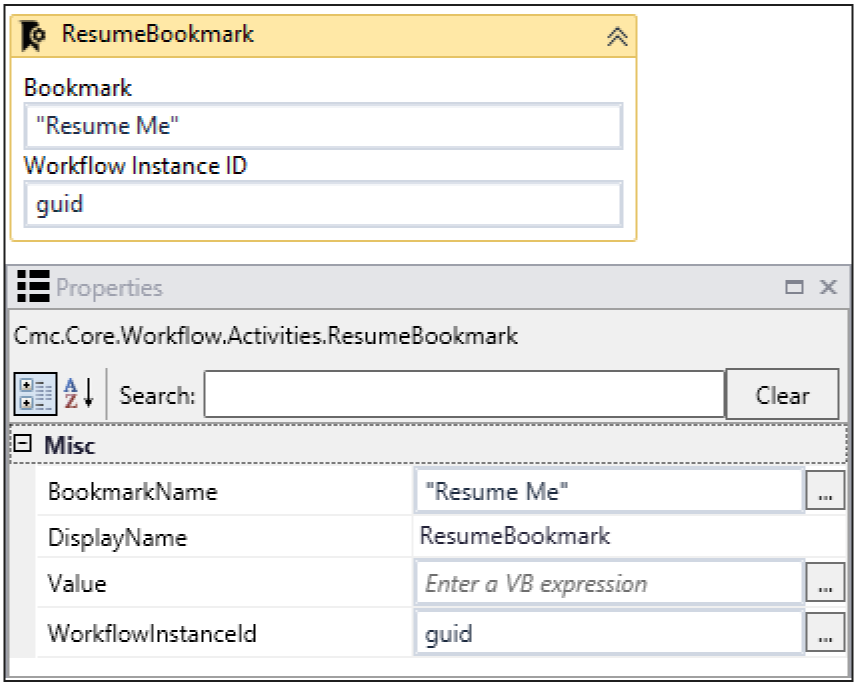
Properties

PostToFacebook Properties

Property	Value	Required	Notes
AccessToken	`InArgument<String>`	Yes	Specify the login for the Facebook page.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Message	`InArgument<String>`	Yes	Specify the message to be posted, for example, `"Check out our Fall Schedule"`
PageId	`InArgument<String>`	Yes	Specify the URL of the Facebook page where the message is to be posted.

ResumeBookmark

The ResumeBookmark activity is used to resume a workflow that has been persisted via the CreateBookmark activity.



ResumeBookmark

Bookmark: "Resume Me"

Workflow Instance ID: guid

Properties

Cmc.Core.Workflow.Activities.ResumeBookmark

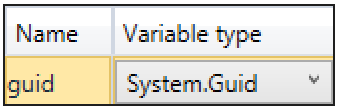
Search:

Misc

BookmarkName	"Resume Me"
DisplayName	ResumeBookmark
Value	*Enter a VB expression*
WorkflowInstanceId	guid

Properties

ResumeBookmark Properties

Property	Value	Required	Notes
BookmarkName	`InArgument<String>`	Yes	Specify the name of the bookmark to resume.
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Value	`InArgument<Object>`	No	Specify an optional argument to pass to the workflow when it resumes.
WorkflowInstanceId	`InArgument<Guid>`	Yes	Specify the Id associated with the workflow instance to resume using a VB expression or variable. The variable type for the InArgument is System.Guid.  Name: guid, Variable type: System.Guid

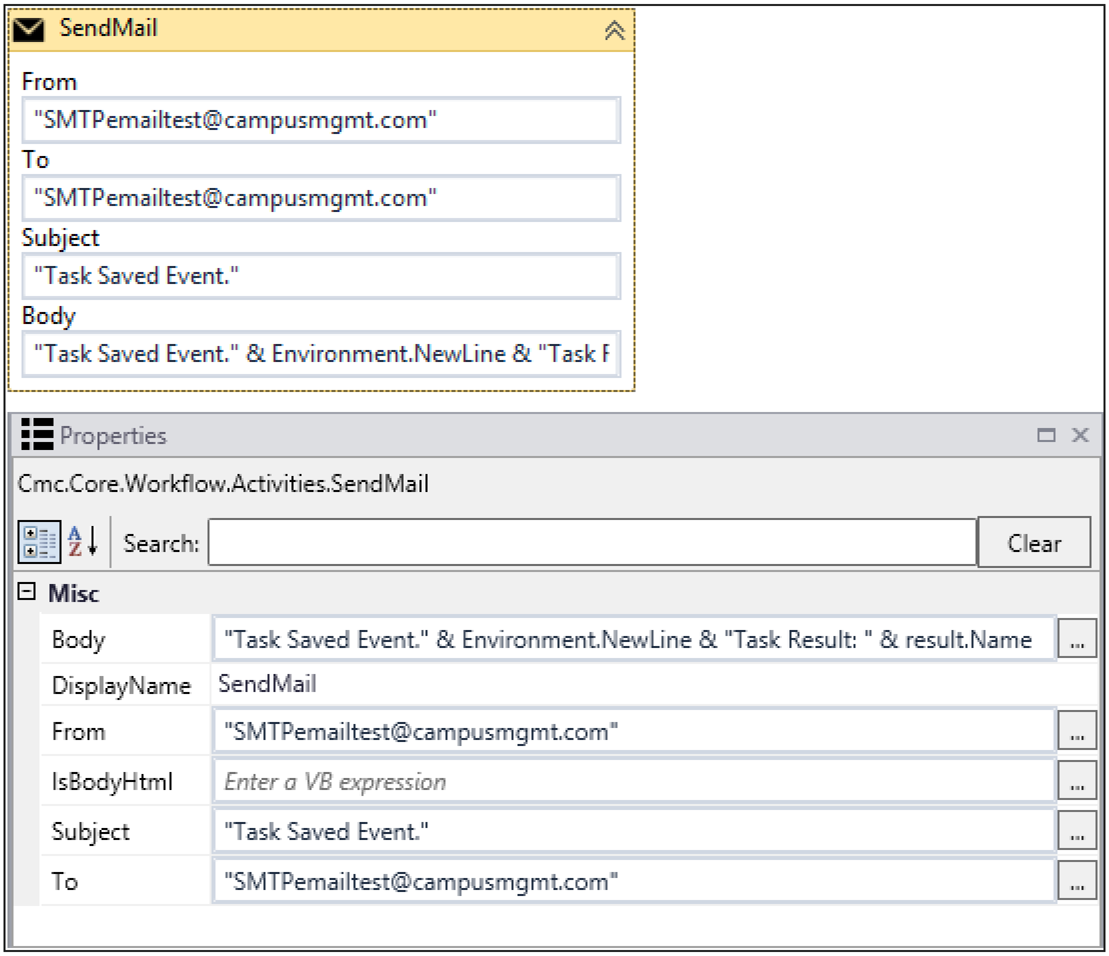
To see how ResumeBookmark can be used in a workflow, refer to:

- Create a Long Running Workflow
- Wake up the Long Running Workflow

SendMail

The SendMail activity enables you to send an email message. The email is sent using the SMTP service defined in the configuration file (`app.config` or `web.config`) of the host where the workflows are installed.

This email service does not use the messaging service that is integrated in Anthology Student. To send email through Anthology Student using the Anthology Student tracking system, use the CreateTask (V2) activity and create Contact Manager task that sends email.



SendMail

From
"SMTPemailtest@campusmgmt.com"
To
"SMTPemailtest@campusmgmt.com"
Subject
"Task Saved Event."
Body
"Task Saved Event." & Environment.NewLine & "Task F

Properties

Cmc.Core.Workflow.Activities.SendMail

Search:

Misc

Body	"Task Saved Event." & Environment.NewLine & "Task Result: " & result.Name
DisplayName	SendMail
From	"SMTPemailtest@campusmgmt.com"
IsBodyHtml	*Enter a VB expression*
Subject	"Task Saved Event."
To	"SMTPemailtest@campusmgmt.com"

Properties

SendMail Properties

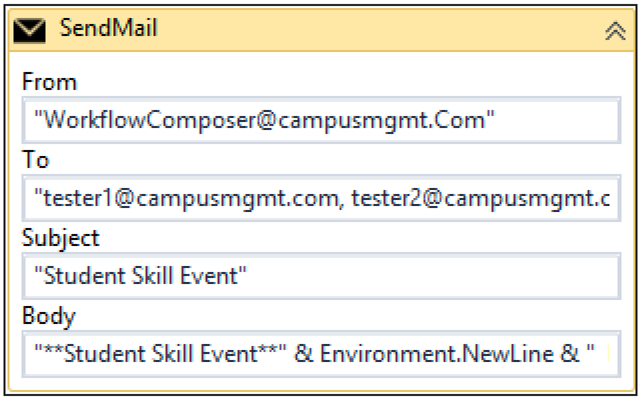
Property	Value	Required	Notes
Body	`InArgument<String>`	Yes	Specify the body text of the message using a VB expression or variable.
DisplayName	`String`	No	Specify a name for the activity or accept the default.

Property	Value	Required	Notes
From	`InArgument<String>`	Yes	Specify the email address of the sender using a VB expression or variable.
IsBodyHtml	`InArgument<Boolean>`	No	Specify whether the body text is formatted in HTML (optional).
Subject	`InArgument<String>`	Yes	Specify the subject of the message using a VB expression or variable.
To	`InArgument<String>`	Yes	Specify the email address of the receiver using a VB expression or variable, for example: `entity.Emails(0).EmailAddress`

SendMail Example

You can use the SendMail activity to notify one or multiple persons of an event. The message can contain any body text, including values that are obtained from other activities in the workflow.

Drag a **SendMail** activity into the sequence and specify the **From**, **To**, **Subject**, and **Body** values.



From
"WorkflowComposer@campusmgmt.Com"

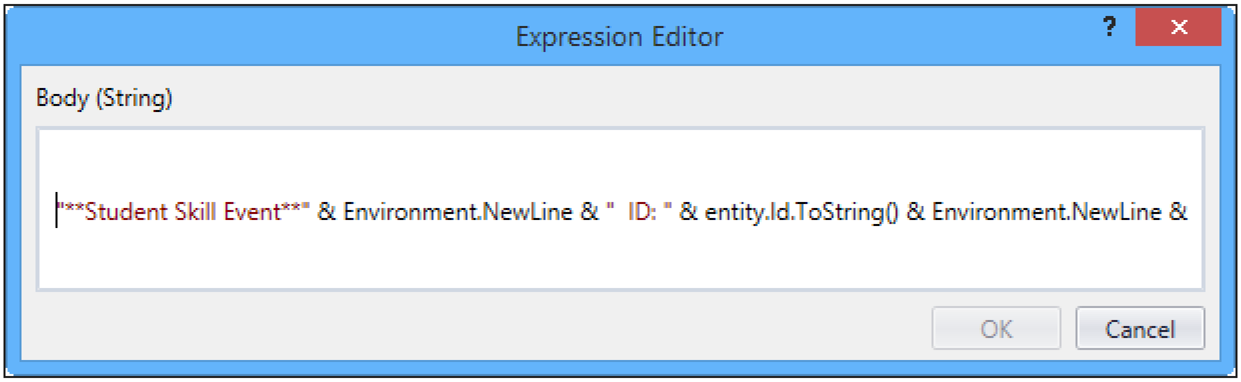
To
"tester1@campusmgmt.com, tester2@campusmgmt.c

Subject
"Student Skill Event"

Body
"**Student Skill Event**" & Environment.NewLine & "

Notes:

- Multiple email addresses, separated by commas, can be specified in the To field.
- In our example the Body field contains a VB expression that lists a number of values obtained from the event, e.g., ID, Skill ID, Student Placement Summary ID, and State. The values are converted to text strings and separated by `Environment.Newline` expressions.



```
"**Student Skill Event**" & Environment.NewLine & "  ID: " & entity.Id.ToString() & Environment.NewLine &
```

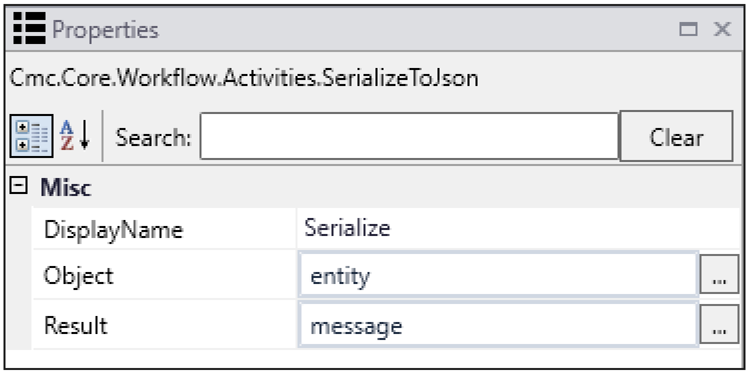
The expression in the Body field is shown here with line breaks for clarity:

```
"**Student Skill Event**" &
Environment.NewLine &
" ID: " &
entity.Id.ToString() &
Environment.NewLine &
" Skill ID: " &
entity.SkillId.ToString() &
Environment.NewLine &
" Student Placement Summary ID: " &
entity.StudentPlacementSummaryId.ToString() &
Environment.NewLine &
Environment.NewLine &
" Other Entity Data" &
Environment.NewLine &
" State: " &
entity.EntityState.ToString()
```

Tip: Use a text editor, e.g., Notepad, to build expressions and paste them into the Expression Editor in Workflow Designer.

SerializeToJson

The SerializeToJson activity initializes a new instance of the SerializeToJson class. The activity serializes an object to JSON.



Cmc.Core.Workflow.Activities.SerializeToJson

Search:

DisplayName	Serialize
Object	entity
Result	message

Properties

SerializeToJson Properties

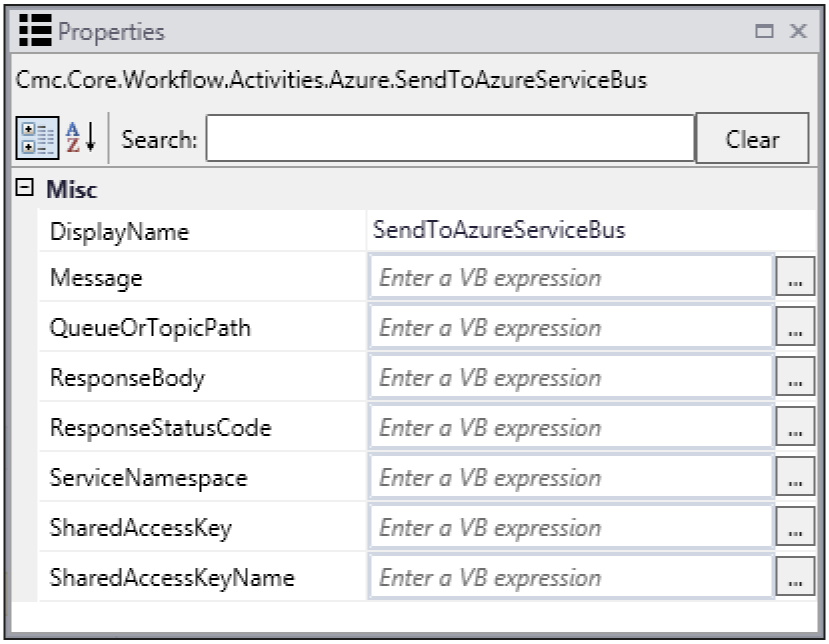
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Object	`InArgument<Object>`	Yes	Specify the object to be serialized.
Result	`OutArgument<String>`	Yes	The JSON string created by this workflow activity. This is a variable that can be used as input for sub-sequent workflow activities.

Cmc.Core.Workflow.Activities.Azure

SendToAzureServiceBus

The SendToAzureServiceBus activity sends messages to the Azure Service Bus. Service bus messages are sent asynchronously, i.e., you place a message on the queue, and at some point a subscriber/listener of that queue will handle the message.

For the SendToAzureServiceBus activity, the workflow logic cannot depend on getting an immediate result from the process –- all you will know is that the message was successfully queued. If you want to get or post data and want to know the result immediately (synchronously), use the Http activity. The Http activity will execute (send) a request and you will get a response from the Url end-point that is being posted to. For more information, see example Http vs. SendToAzureServiceBus.



Cmc.Core.Workflow.Activities.Azure.SendToAzureServiceBus

DisplayName	SendToAzureServiceBus
Message	Enter a VB expression
QueueOrTopicPath	Enter a VB expression
ResponseBody	Enter a VB expression
ResponseStatusCode	Enter a VB expression
ServiceNamespace	Enter a VB expression
SharedAccessKey	Enter a VB expression
SharedAccessKeyName	Enter a VB expression

Properties

SendToAzureServiceBus Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Message	`InArgument<String>`	Yes	The message sent to the Azure Service Bus.
QueueOrTopicPath	`InArgument<String>`	Yes	The queue or topic path for the Azure Service Bus.
ResponseBody	`OutArgument<String>`	No	The response body returned from the server.

Property	Value	Required	Notes
ResponseStatusCode	`OutArgument<HttpStatusCode>`	No	The response status code. Represents the HTTP response status code issued by the server in response to the request (e.g., 200, 401, 500, etc.). Initializes a new instance of the Http class.
ServiceNamespace	`InArgument<String>`	No	The service namespace. See Note.
SharedAccessKey	`InArgument<String>`	No	The shared access key. See Note.
SharedAccessKeyName	`InArgument<String>`	No	The name of the shared access key. See Note.

Note:

The properties ServiceBusNamespace, SharedAccessKey, and SharedAccessKeyName are not required; however, if they are not provided, the activity will pull these settings from the web.config appSettings section. This is to allow workflows to be reused from environment to environment without modification.

Below is an example of the web.config app settings:

```
<add key="azureServiceBus:serviceNamespace" value="nexus-student-integration-bus" />

<add key="azureServiceBus:sharedAccessKeyName" value="SendSharedAccessKey" />

<add key="azureServiceBus:sharedAccessKey" value="ZRXqcCfXQaGMi0FXTp6iNtFjMXKG+adnZTO3CcNAqDA="
/>
```

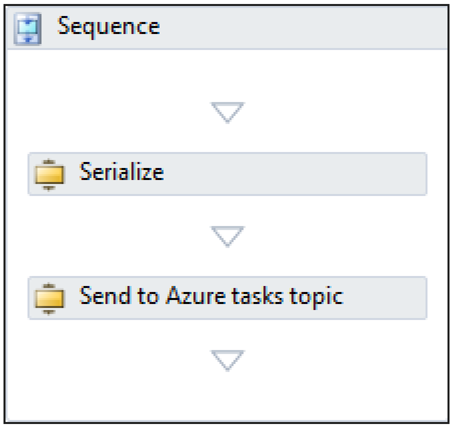
⚠ Clients using their own Azure Subscription (customer side) need to specify the Service Bus settings applicable to their environment.

Examples

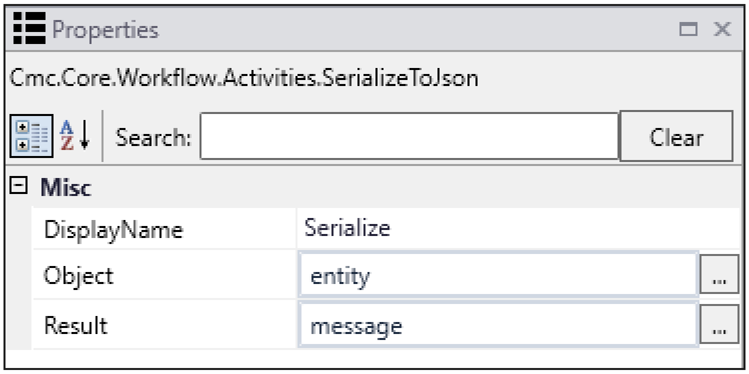
Send Message

The following workflow example shows how the activity can be used to send messages to the Azure Service Bus.

The workflow Cmc.Nexus.Crm.Entities.TaskEntity_SavedEvent_Sample%20-%20Azure%20Service%20Bus.xaml is available on GitHub.



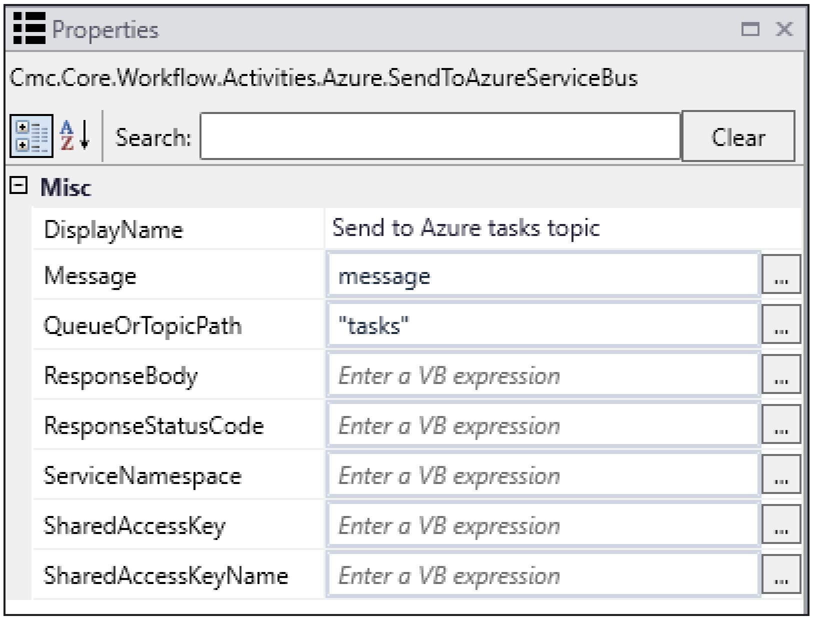
1. The SerializeToJson activity serializes an input argument object named "entity" and produces the output string named "message".



Cmc.Core.Workflow.Activities.SerializeToJson

DisplayName	Serialize
Object	entity
Result	message

2. The SendToAzureServiceBus activity uses the serialized "message" string as input argument and creates the output argument named "tasks".



Cmc.Core.Workflow.Activities.Azure.SendToAzureServiceBus

DisplayName	Send to Azure tasks topic
Message	message
QueueOrTopicPath	"tasks"
ResponseBody	*Enter a VB expression*
ResponseStatusCode	*Enter a VB expression*
ServiceNamespace	*Enter a VB expression*
SharedAccessKey	*Enter a VB expression*
SharedAccessKeyName	*Enter a VB expression*

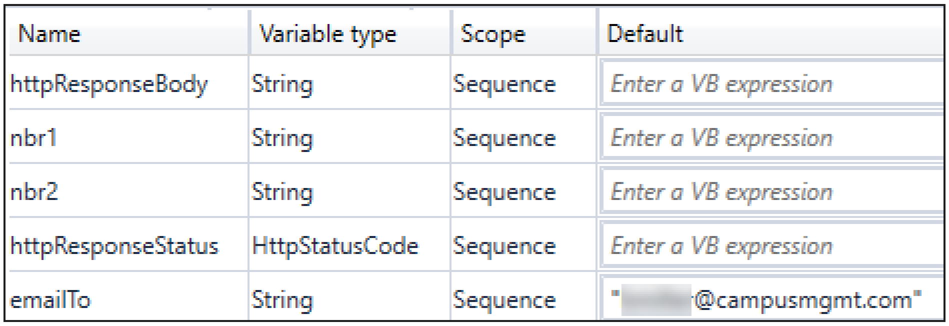
Http vs. SendToAzureServiceBus

To demonstrate the difference between the Http and SendToAzureServiceBus activities, we created a workflow that multiplies two numbers (24 x 365) and returns the result (8,760).

- The Http activity returns the result immediately to a workflow variable.
- The SendToAzureServiceBus activity sends the result to the service bus where it is processed by an application that is listening for messages. Then response is sent to the email address specified in the request.

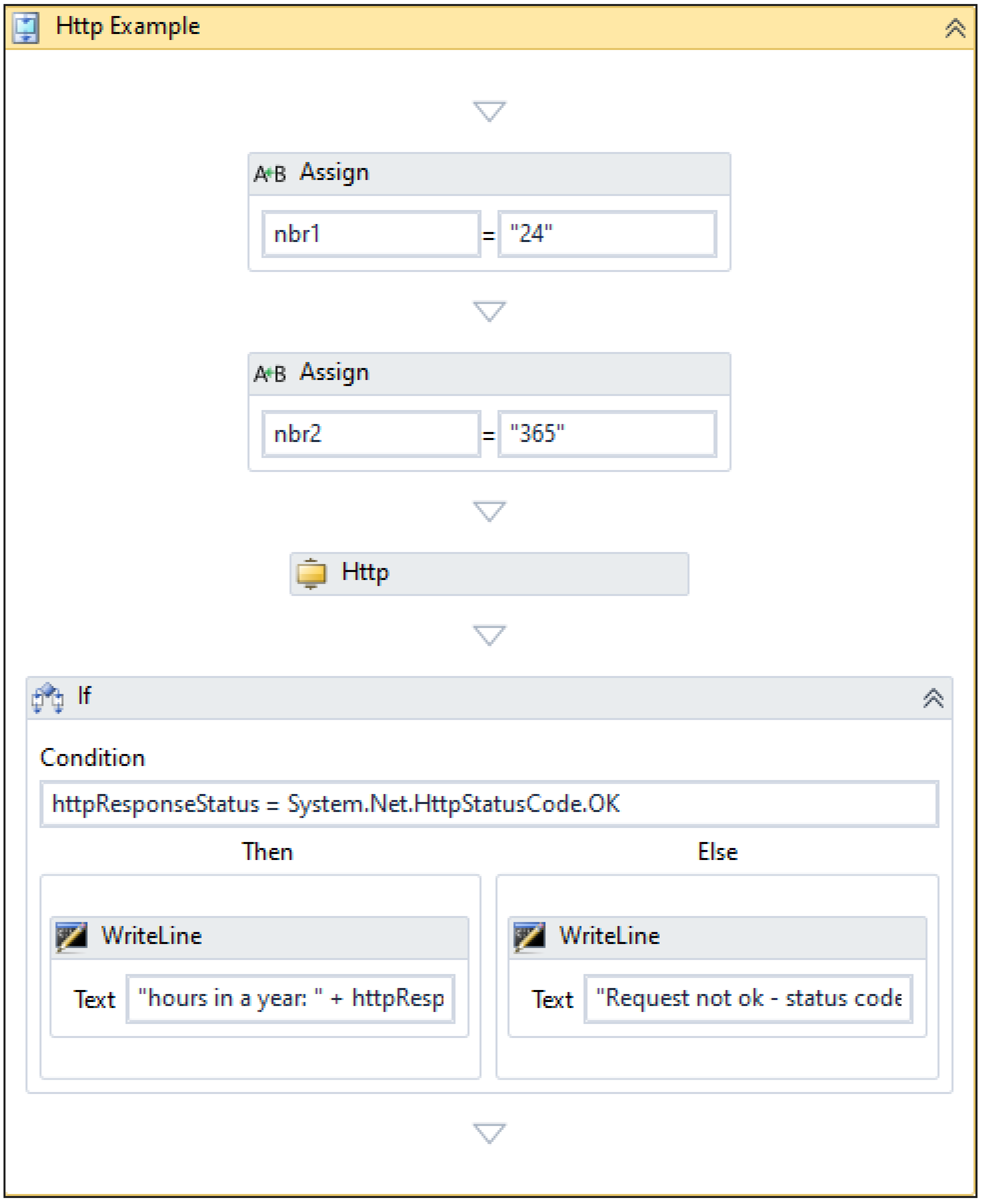
  In more complex scenarios, the response from the service bus listener process could be a call back into another system –- or the service bus listener would forward the message to a 3rd party application to be posted to that system.

The workflow uses the following variables:



Name	Variable type	Scope	Default
httpResponseBody	String	Sequence	*Enter a VB expression*
nbr1	String	Sequence	*Enter a VB expression*
nbr2	String	Sequence	*Enter a VB expression*
httpResponseStatus	HttpStatusCode	Sequence	*Enter a VB expression*
emailTo	String	Sequence	"[redacted]@campusmgmt.com"

Http Activity



nbr1 = "24"

nbr2 = "365"

Condition

httpResponseStatus = System.Net.HttpStatusCode.OK

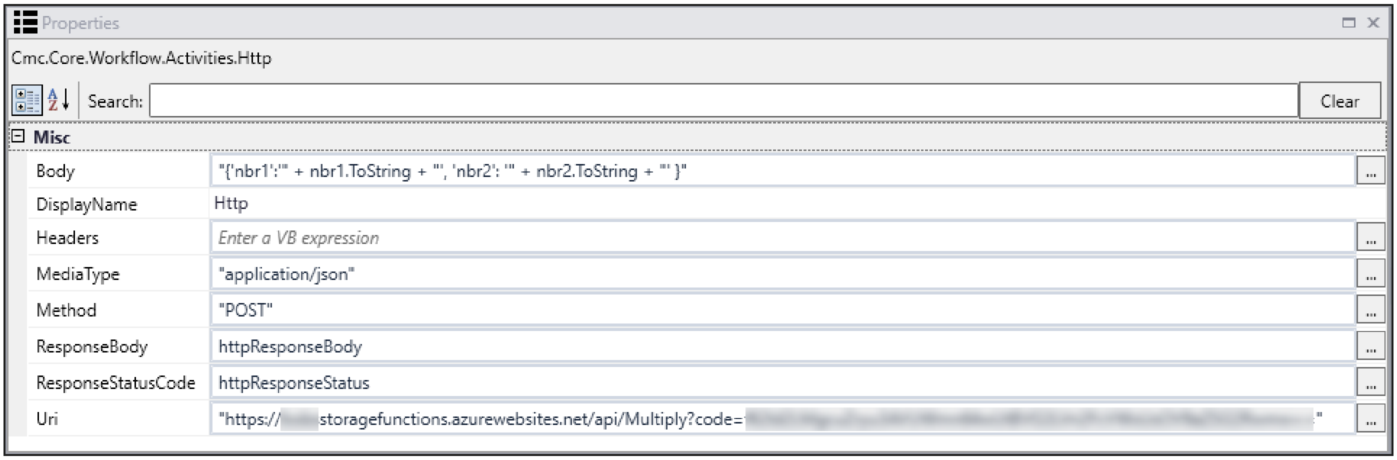
Text "hours in a year: " + httpResp

Text "Request not ok - status code

The Http activity:

- Uses the string assignments (nbr1 and nbr2) as input arguments in the Body property.
- Defines the input as MediaType = "application/json".
- Invokes the "POST" method.
- Creates the output argument named "httpResponseBody".
- Creates the output argument named "httpResponseStatus" whose value is checked in the If Condition.
- Sends the output to a Uri on an Azure web site that hosts an API.

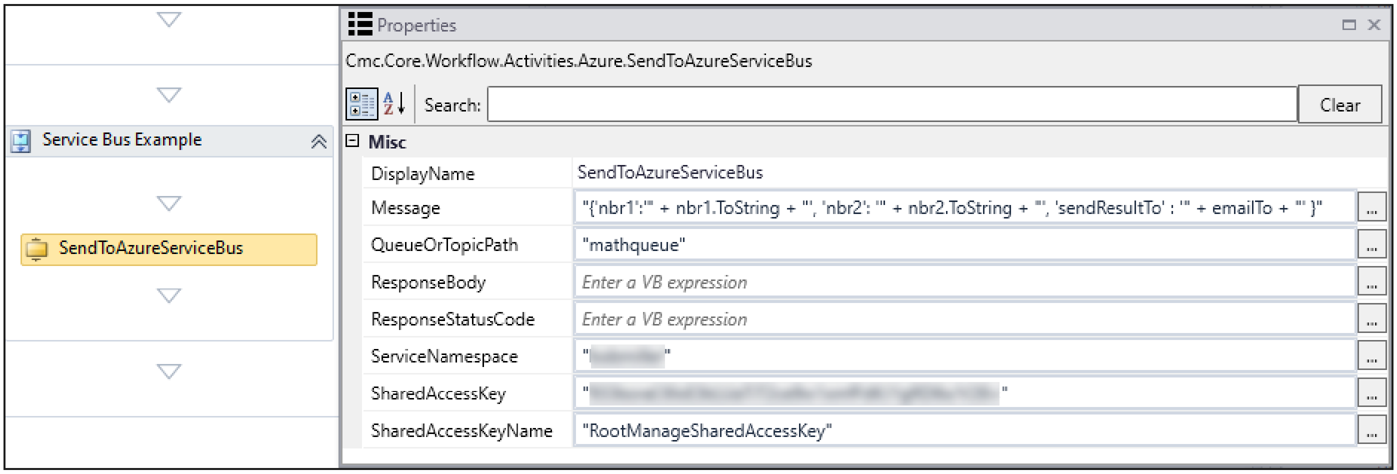
The API multiplies the numbers 2 numbers in the request Body (nbr1 and nbr2) and returns the result.



Cmc.Core.Workflow.Activities.Http

Body	"{'nbr1':'" + nbr1.ToString + "', 'nbr2': '" + nbr2.ToString + "' }"
DisplayName	Http
Headers	*Enter a VB expression*
MediaType	"application/json"
Method	"POST"
ResponseBody	httpResponseBody
ResponseStatusCode	httpResponseStatus
Uri	"https://[redacted]storagefunctions.azurewebsites.net/api/Multiply?code=[redacted]"

SendToAzureServiceBus Activity



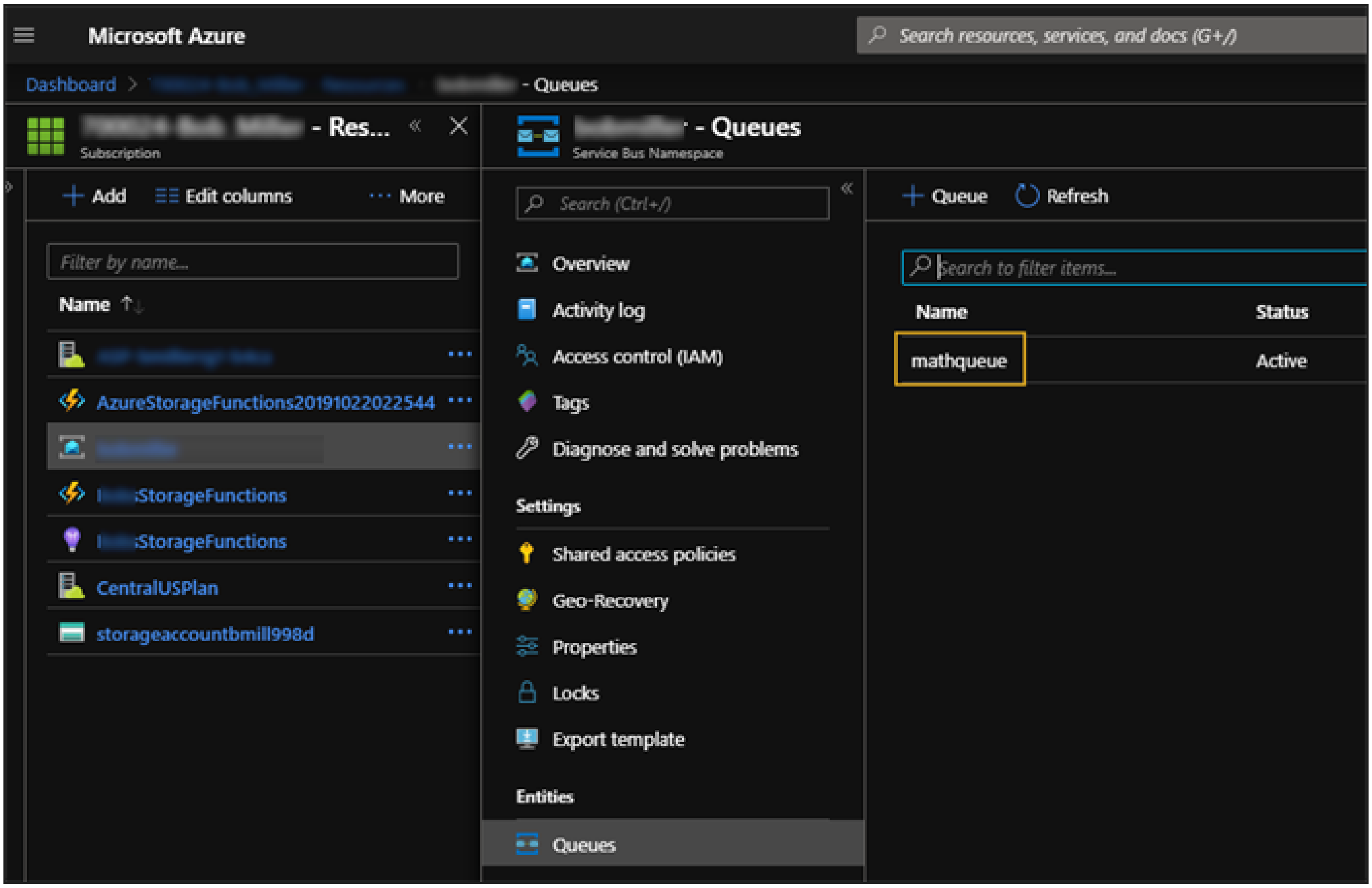
Cmc.Core.Workflow.Activities.Azure.SendToAzureServiceBus

DisplayName	SendToAzureServiceBus
Message	"{'nbr1':'" + nbr1.ToString + "', 'nbr2': '" + nbr2.ToString + "', 'sendResultTo' : '" + emailTo + "' }"
QueueOrTopicPath	"mathqueue"
ResponseBody	*Enter a VB expression*
ResponseStatusCode	*Enter a VB expression*
ServiceNamespace	"[redacted]"
SharedAccessKey	"[redacted] "
SharedAccessKeyName	"RootManageSharedAccessKey"

The SendToAzureServiceBus activity:

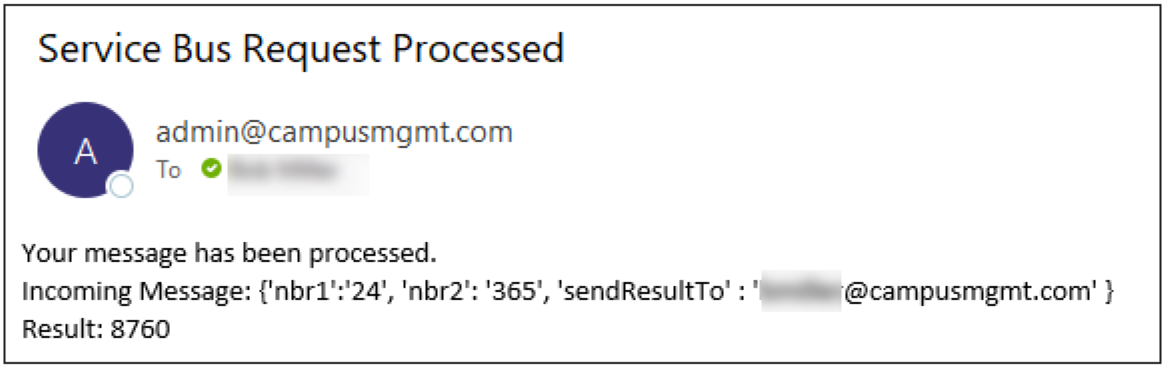
- Sends the string assignments (nbr1 and nbr2) and "emailTo" variable to the Azure Service Bus.
- Specifies the path for the Azure Service Bus as "mathqueue".
- Specifies the user's service name space and access key in Azure.

In Azure, the message is placed in the "mathqueue" and processed.



Name	Status
mathqueue	Active

When the service bus request is processed, an email is sent to the user.



Service Bus Request Processed

admin@campusmgmt.com
To [redacted]

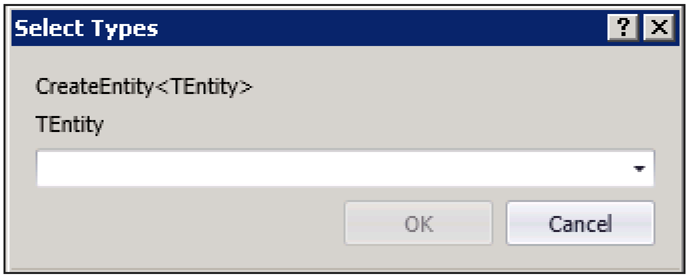
Your message has been processed.
Incoming Message: {'nbr1':'24', 'nbr2': '365', 'sendResultTo' : '[redacted]@campusmgmt.com' }
Result: 8760

Cmc.Core.Workflow.Activities.EntityModel

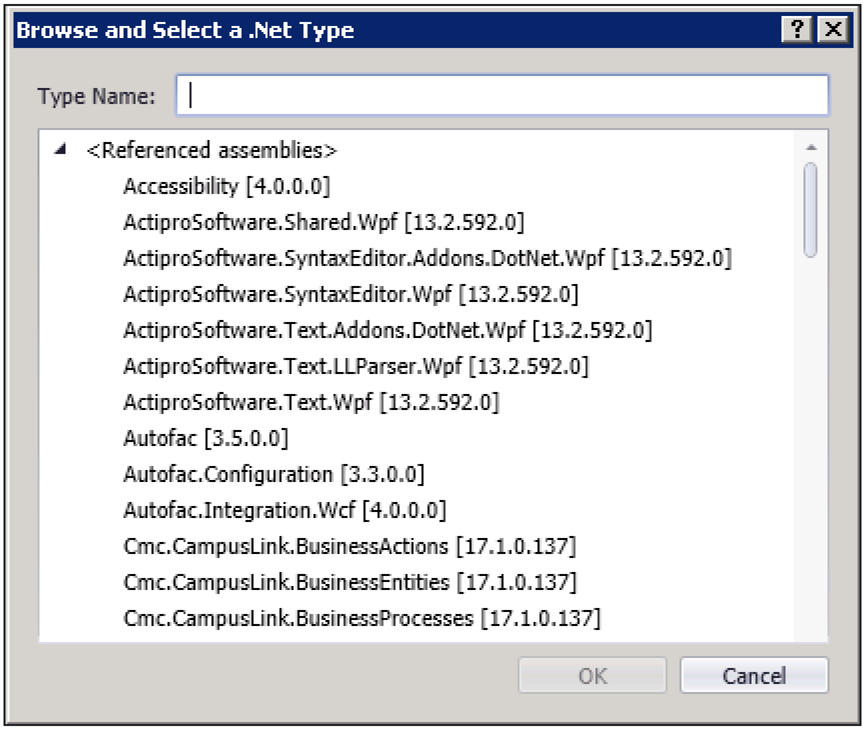
CreateEntity<>

The CreateEntity<> activity invokes the New method of an entity service to create an instance of an entity. To save the instance of the created entity, use the SaveEntity<> activity.

When you drag the CreateEntity<> activity into the Designer window, you are prompted to select the entity type (TEntity).

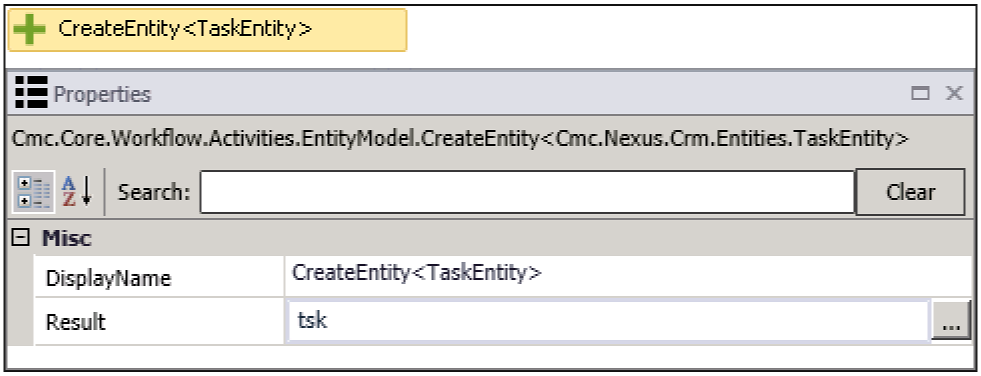


When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.



<Referenced assemblies>
Accessibility [4.0.0.0]
ActiproSoftware.Shared.Wpf [13.2.592.0]
ActiproSoftware.SyntaxEditor.Addons.DotNet.Wpf [13.2.592.0]
ActiproSoftware.SyntaxEditor.Wpf [13.2.592.0]
ActiproSoftware.Text.Addons.DotNet.Wpf [13.2.592.0]
ActiproSoftware.Text.LLParser.Wpf [13.2.592.0]
ActiproSoftware.Text.Wpf [13.2.592.0]
Autofac [3.5.0.0]
Autofac.Configuration [3.3.0.0]
Autofac.Integration.Wcf [4.0.0.0]
Cmc.CampusLink.BusinessActions [17.1.0.137]
Cmc.CampusLink.BusinessEntities [17.1.0.137]
Cmc.CampusLink.BusinessProcesses [17.1.0.137]

After you have selected an entity, the name of the entity is inserted into the DisplayName field, e.g., CreateEntity<TaskEntity>. Proceed to specify the Result.



CreateEntity<TaskEntity>

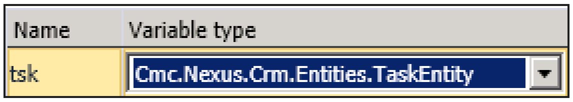
Cmc.Core.Workflow.Activities.EntityModel.CreateEntity<Cmc.Nexus.Crm.Entities.TaskEntity>

Misc

DisplayName	CreateEntity<TaskEntity>
Result	tsk

Properties

CreateEntity<> Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Result	`OutArgument<Entity>`	Yes	The entity created by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to the entity that matches the previously selected entity type, for example, **Cmc.Nexus.Crm.Entities.TaskEntity** and click **OK**.  Name: tsk Variable type: Cmc.Nexus.Crm.Entities.TaskEntity

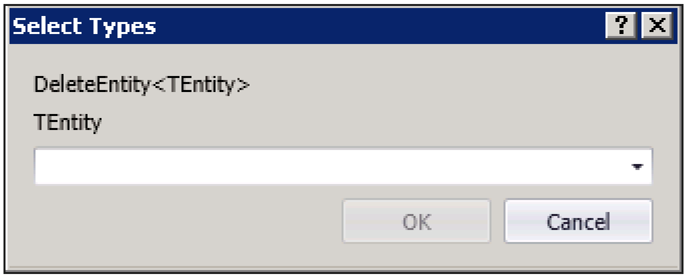
If you are working with the ApplicantEntity in Anthology Student, refer to Create/Save ApplicantEntity and Update Derived Fields.

DeleteEntity<>

The DeleteEntity<> activity invokes the Delete method of an entity service to delete an instance of an entity.

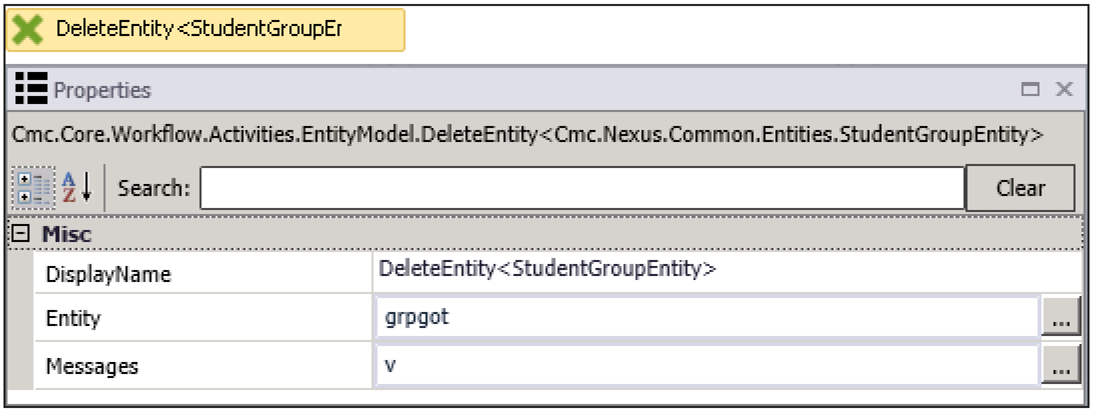
Note: The DeleteEntity<> activity does not support the deletion of CampusNexus CRM entities.

When you drag the DeleteEntity<> activity into the Designer window, you are prompted to select the entity type.



When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.

After you have selected an entity, the name of the entity is inserted into the DisplayName field, e.g., DeleteEntity<StudentGroupEntity>. Proceed to specify the entity to be deleted and, optionally, a validation message.



Cmc.Core.Workflow.Activities.EntityModel.DeleteEntity<Cmc.Nexus.Common.Entities.StudentGroupEntity>

DisplayName	DeleteEntity<StudentGroupEntity>
Entity	grpgot
Messages	v

Properties

DeleteEntity<> Properties

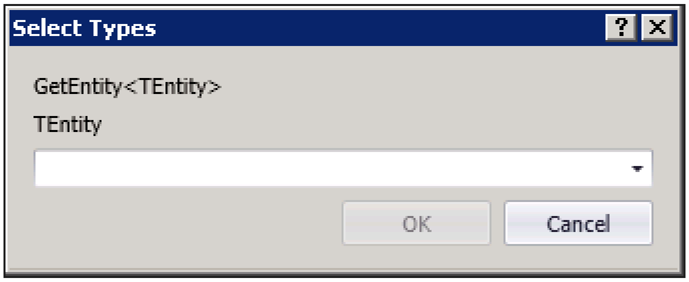
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Entity	`InOutArgument<Entity>`	Yes	Specify the entity previously retrieved with a GetEntity<> activity using a VB expression or variable.

Property	Value	Required	Notes
Messages	`InArgument <ICollection <ValidationMessage>>`	No	Specify the validation message to be displayed when the entity is deleted. Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

GetEntity<>

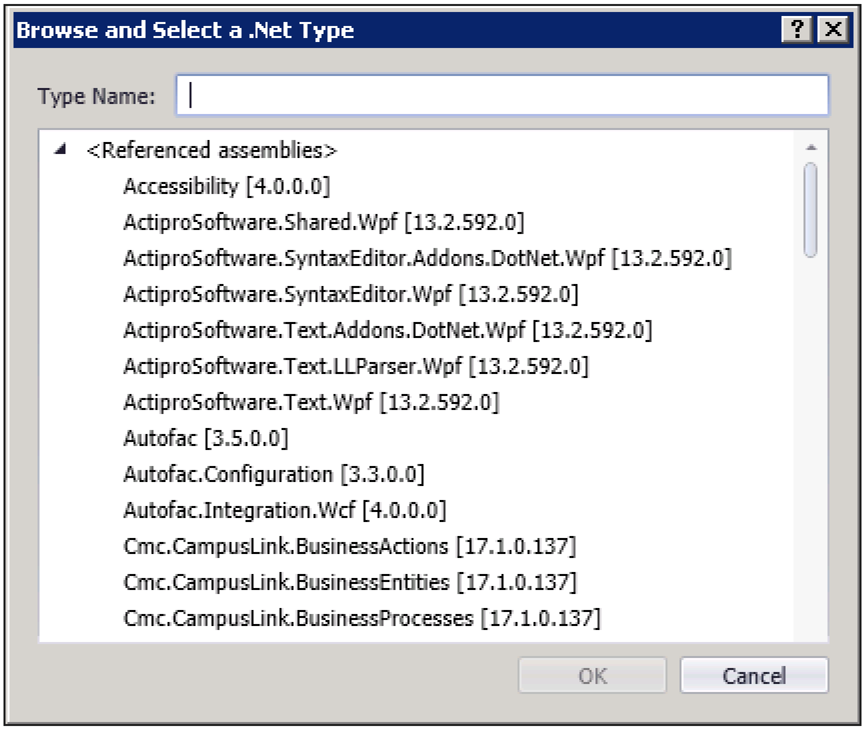
The GetEntity<> activity invokes the Get method of an entity service to retrieve an instance of an entity.

When you drag the GetEntity<> activity into the Designer window, you are prompted to select the entity type (TEntity).



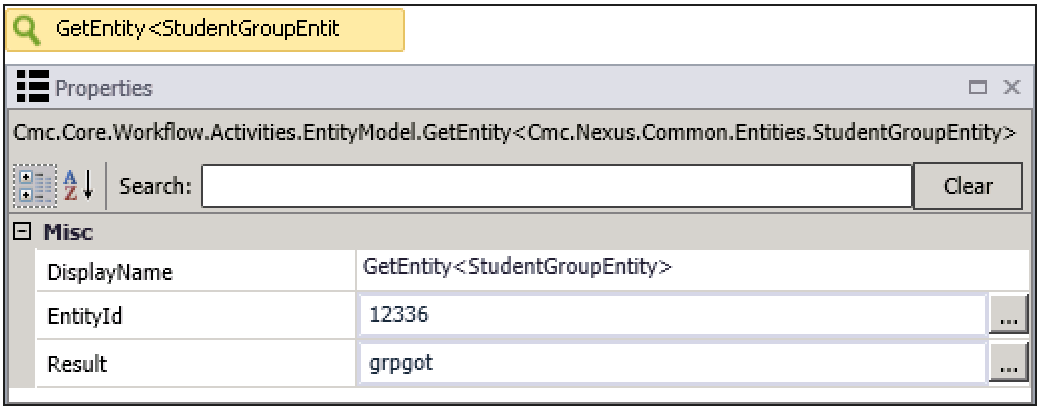
When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.

Note: The GetEntity<> activity is not supported for the *StudentAdvisors* entity because the primary key for this entity consists of two properties. To work with the StudentAdvisors entity, use the StudentAdvisorService – GetStudentAdvisors operation.



◢ <Referenced assemblies>
- Accessibility [4.0.0.0]
- ActiproSoftware.Shared.Wpf [13.2.592.0]
- ActiproSoftware.SyntaxEditor.Addons.DotNet.Wpf [13.2.592.0]
- ActiproSoftware.SyntaxEditor.Wpf [13.2.592.0]
- ActiproSoftware.Text.Addons.DotNet.Wpf [13.2.592.0]
- ActiproSoftware.Text.LLParser.Wpf [13.2.592.0]
- ActiproSoftware.Text.Wpf [13.2.592.0]
- Autofac [3.5.0.0]
- Autofac.Configuration [3.3.0.0]
- Autofac.Integration.Wcf [4.0.0.0]
- Cmc.CampusLink.BusinessActions [17.1.0.137]
- Cmc.CampusLink.BusinessEntities [17.1.0.137]
- Cmc.CampusLink.BusinessProcesses [17.1.0.137]

After you have selected an entity, the name of the entity is inserted into the DisplayName field, e.g., GetEntity<StudentGroupEntity>. Proceed to specify the EntityId and Result.



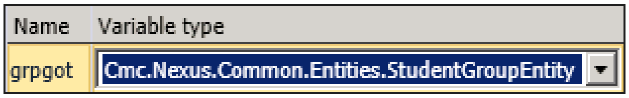
GetEntity<StudentGroupEntit

Cmc.Core.Workflow.Activities.EntityModel.GetEntity<Cmc.Nexus.Common.Entities.StudentGroupEntity>

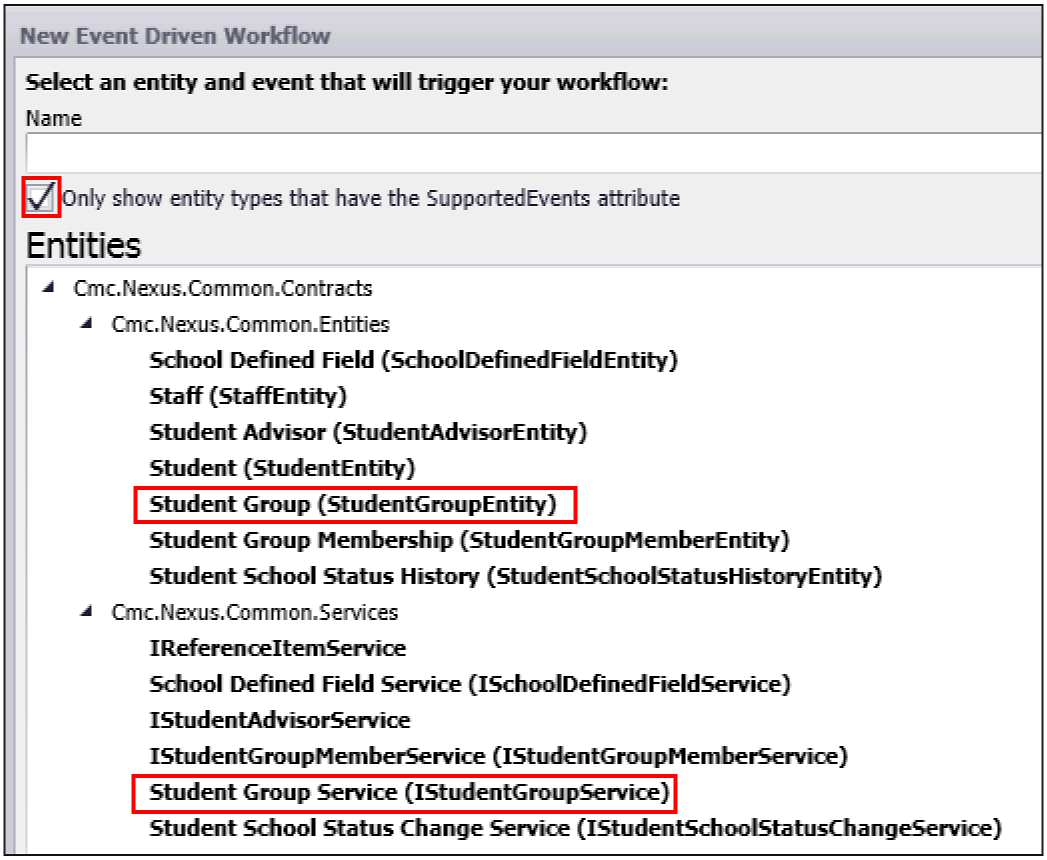
DisplayName	GetEntity<StudentGroupEntity>
EntityId	12336
Result	grpgot

Properties

GetEntity<> Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
EntityId	`InArgument<Int32>`	Yes	Specify the entity identifier using a VB expression or variable.
Result	`OutArgument<Entity>`	Yes	The entity retrieved by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to the entity that matches the previously selected entity type, for example, **Cmc.Nexus.Common.Entities.StudentGroupEntity** and click **OK**.  Name: grpgot Variable type: Cmc.Nexus.Common.Entities.StudentGroupEntity

Note: Before you use this activity, make sure that both the **entity** you want to work with and a matching **service** are available and enabled in Workflow Designer. To check this, click **New Event Workflow**, select the filter *"Only show entity types that have the SupportedEvents attribute"*, and locate the entity type, for example, Student Group. In this case Workflow Designer shows that both the *Student Group (StudentGroupEntity)* and the corresponding *Student Group Service (iStudentGroupService)* are enabled. This indicates that the GetEntity activity is supported for the selected entity type.



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

Entities

- Cmc.Nexus.Common.Contracts
  - Cmc.Nexus.Common.Entities
    - School Defined Field (SchoolDefinedFieldEntity)
    - Staff (StaffEntity)
    - Student Advisor (StudentAdvisorEntity)
    - Student (StudentEntity)
    - Student Group (StudentGroupEntity)
    - Student Group Membership (StudentGroupMemberEntity)
    - Student School Status History (StudentSchoolStatusHistoryEntity)
  - Cmc.Nexus.Common.Services
    - IReferenceItemService
    - School Defined Field Service (ISchoolDefinedFieldService)
    - IStudentAdvisorService
    - IStudentGroupMemberService (IStudentGroupMemberService)
    - Student Group Service (IStudentGroupService)
    - Student School Status Change Service (IStudentSchoolStatusChangeService)

GetEntityCollection<>

Prerequisites

The GetEntityCollection<> activity is available in Workflow Composer version 2.7 and later and requires the following **minimum** versions of activities and contracts:

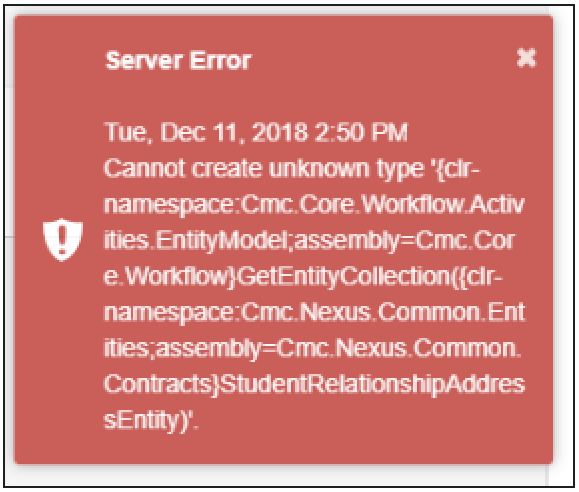
- Anthology Student version 20.0.x

  — OR —

- CampusNexus CRM version 12.2.x

The minimum Cmc.Core.dll version installed in Program Files (x86)\CMC\Workflow must be 5.1.167 or greater.

Note: If you use the activity with Student 19.0 and Workflow Composer 2.7, you won’t see any errors in Workflow Composer (because it has minimum Cmc.Core.dll version), but you’ll see a server error at runtime.



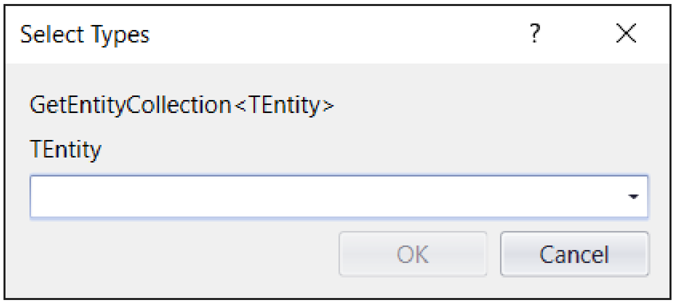
Server Error

Tue, Dec 11, 2018 2:50 PM
Cannot create unknown type '{clr-namespace:Cmc.Core.Workflow.Activities.EntityModel;assembly=Cmc.Core.Workflow}GetEntityCollection({clr-namespace:Cmc.Nexus.Common.Entities;assembly=Cmc.Nexus.Common.Contracts}StudentRelationshipAddressEntity)'.

Purpose

The GetEntityCollection<> activity provides the ability to retrieve a collection of values (i.e., rows in a database table) for a given entity by passing in an array of Ids. The activity returns an array of entities in the “Entities” output argument.

When you drag the GetEntityCollection<> activity into the Designer window, you are prompted to select the entity type (TEntity).

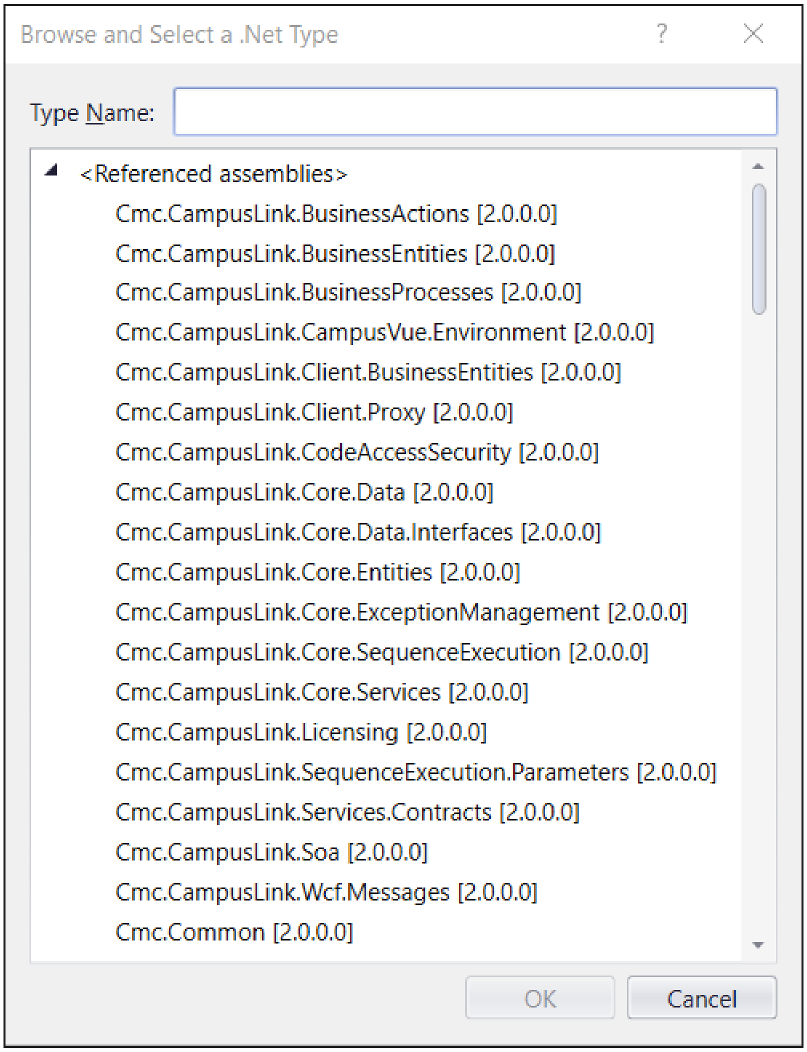


Select Types

GetEntityCollection<TEntity>

TEntity

When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.



Type Name:

- <Referenced assemblies>
  - Cmc.CampusLink.BusinessActions [2.0.0.0]
  - Cmc.CampusLink.BusinessEntities [2.0.0.0]
  - Cmc.CampusLink.BusinessProcesses [2.0.0.0]
  - Cmc.CampusLink.CampusVue.Environment [2.0.0.0]
  - Cmc.CampusLink.Client.BusinessEntities [2.0.0.0]
  - Cmc.CampusLink.Client.Proxy [2.0.0.0]
  - Cmc.CampusLink.CodeAccessSecurity [2.0.0.0]
  - Cmc.CampusLink.Core.Data [2.0.0.0]
  - Cmc.CampusLink.Core.Data.Interfaces [2.0.0.0]
  - Cmc.CampusLink.Core.Entities [2.0.0.0]
  - Cmc.CampusLink.Core.ExceptionManagement [2.0.0.0]
  - Cmc.CampusLink.Core.SequenceExecution [2.0.0.0]
  - Cmc.CampusLink.Core.Services [2.0.0.0]
  - Cmc.CampusLink.Licensing [2.0.0.0]
  - Cmc.CampusLink.SequenceExecution.Parameters [2.0.0.0]
  - Cmc.CampusLink.Services.Contracts [2.0.0.0]
  - Cmc.CampusLink.Soa [2.0.0.0]
  - Cmc.CampusLink.Wcf.Messages [2.0.0.0]
  - Cmc.Common [2.0.0.0]

Note that the in and out arguments for the activity are of type **ICollection**.

Properties

GetEntityCollection<> Properties

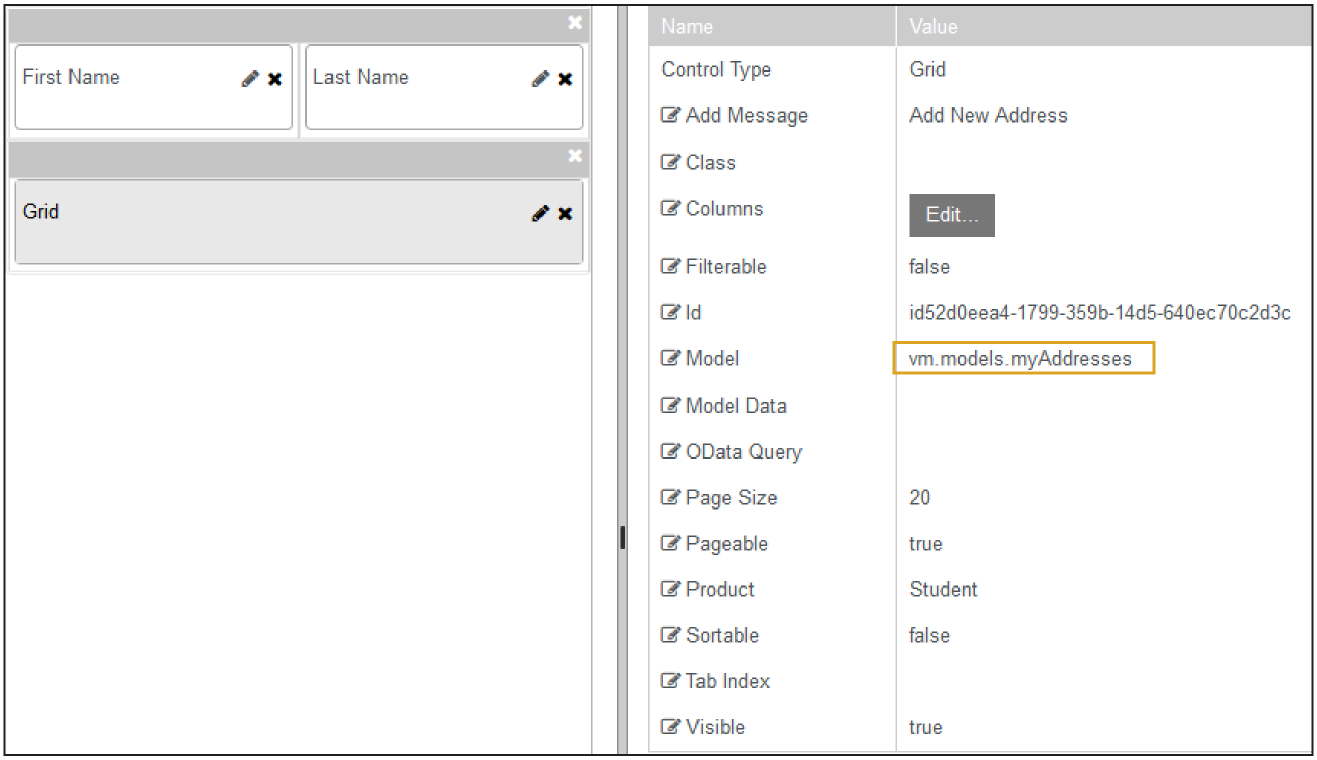
Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Entities	`OutArgument<ICollection<Entity>>`	Yes	Specify the entity array using a VB expression or variable.

Property	Value	Required	Notes
EntityIds	`InArgument<ICollection<Int32>>`	Yes	Specify the entity identifier array using a VB expression or variable.
ResponseItems	`OutArgument<ICollection <EntityServiceResponse<Entity>>>`	No	This is an optional output argument for items retrieved by the activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to the entity that matches the previously selected entity type and click **OK**.
ValidationMessages	`InArgument <ICollection<ValidationMessage>>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

Get/Save EntityCollection Example

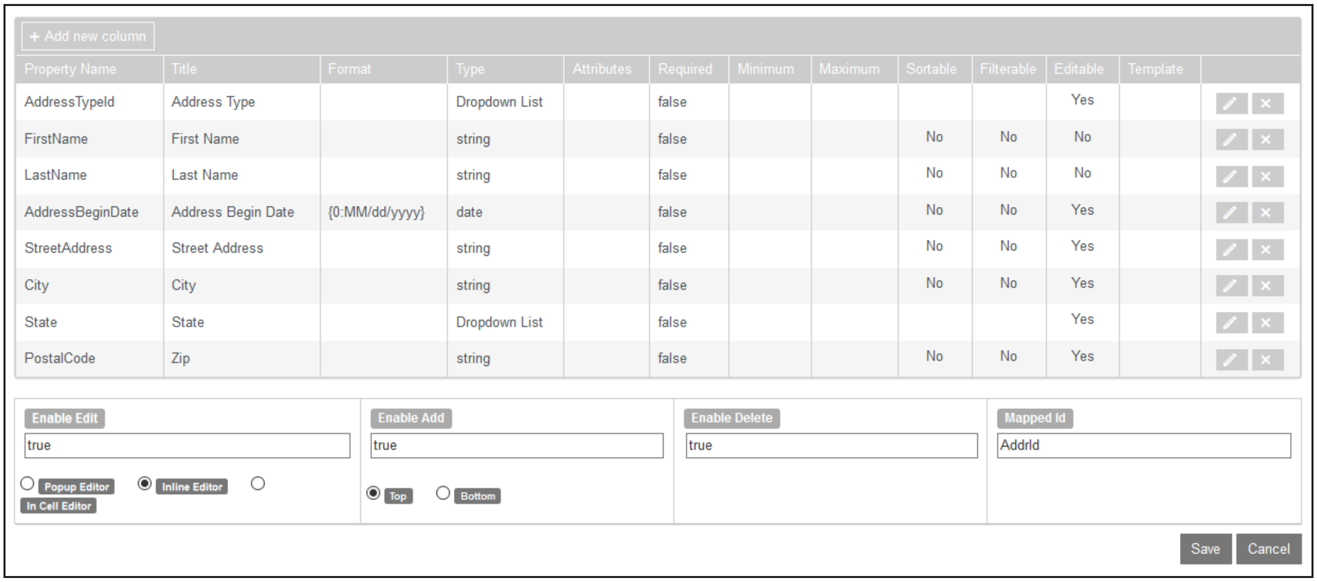
This workflow example is associated with a Forms Builder sequence that retrieves a collection of records for the StudentRelationshipAddressEntity and exposes the records in a grid control. The user of the form sequence is allowed to add and edit data in the grid. The new and modified records are saved to the database.

1. In Form Designer, create a form using the **Grid** component.
2. Bind the Grid component to the workflow using the **Model** property value **vm.models.myAddresses**.



Name	Value
Control Type	Grid
Add Message	Add New Address
Class	
Columns	Edit...
Filterable	false
Id	id52d0eea4-1799-359b-14d5-640ec70c2d3c
Model	vm.models.myAddresses
Model Data	
OData Query	
Page Size	20
Pageable	true
Product	Student
Sortable	false
Tab Index	
Visible	true

3. Configure the **Columns** property to allow the user to add, edit, and delete data.



Property Name	Title	Format	Type	Attributes	Required	Minimum	Maximum	Sortable	Filterable	Editable	Template
AddressTypeId	Address Type		Dropdown List		false					Yes	
FirstName	First Name		string		false			No	No	No	
LastName	Last Name		string		false			No	No	No	
AddressBeginDate	Address Begin Date	{0:MM/dd/yyyy}	date		false			No	No	Yes	
StreetAddress	Street Address		string		false			No	No	Yes	
City	City		string		false			No	No	Yes	
State	State		Dropdown List		false					Yes	
PostalCode	Zip		string		false			No	No	Yes	

Enable Edit: true
○ Popup Editor ● Inline Editor ○ In Cell Editor

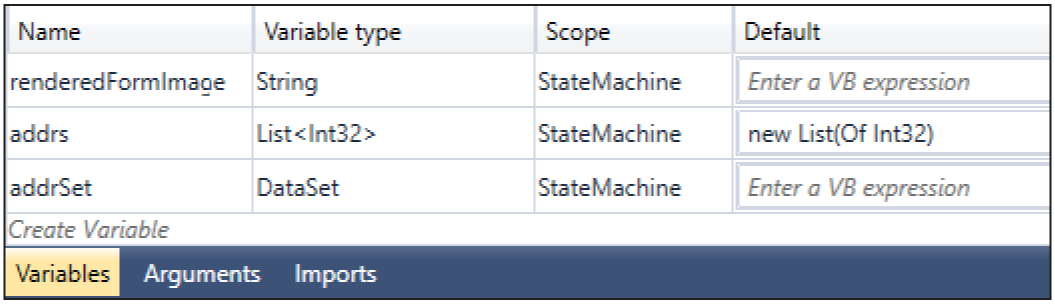
Enable Add: true
● Top ○ Bottom

Enable Delete: true

Mapped Id: AddrId

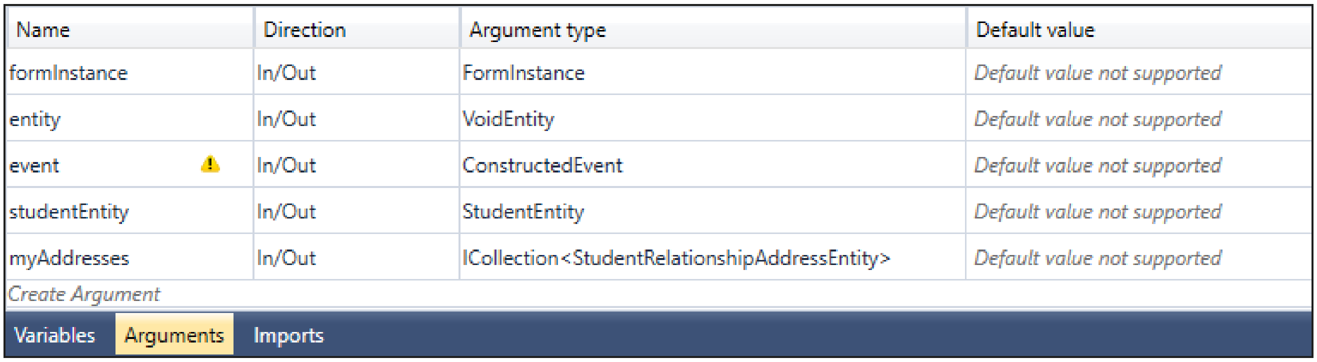
4. In Sequence Designer, add the form to a sequence and open the workflow for the sequence.

5. In Workflow Composer, create the variables shown below.



Name	Variable type	Scope	Default
renderedFormImage	String	StateMachine	*Enter a VB expression*
addrs	List<Int32>	StateMachine	new List(Of Int32)
addrSet	DataSet	StateMachine	*Enter a VB expression*

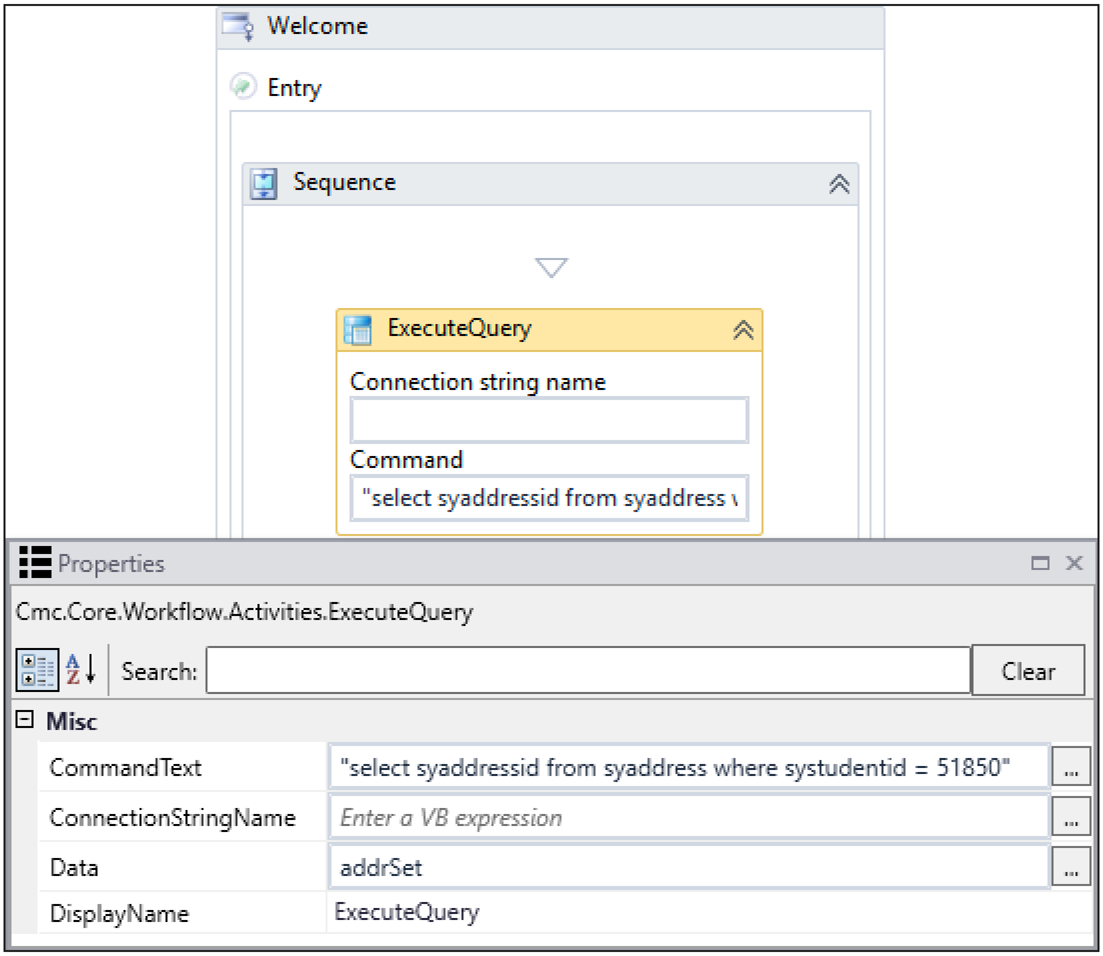
6. Create an argument of type **ICollection<StudentRelationshipAddressEntity>** for the **myAddresses** model value that binds the grid to the workflow. The path to browse to the argument type is: System.Collections.Generic.ICollection<Cmc.Nexus.Common.Entities.StudentRelationshipAddressEntity>.



Name	Direction	Argument type	Default value
formInstance	In/Out	FormInstance	*Default value not supported*
entity	In/Out	VoidEntity	*Default value not supported*
event ⚠	In/Out	ConstructedEvent	*Default value not supported*
studentEntity	In/Out	StudentEntity	*Default value not supported*
myAddresses	In/Out	ICollection<StudentRelationshipAddressEntity>	*Default value not supported*

7. The GetEntityCollection<> activity needs a list of ids for the collection of the same entity type to retrieve. To achieve this, drag an ExecuteQuery activity into the Entry section of the Welcome form. This activity retrieves a set of document ids for a student from the database and returns the data in a variable named **addrSet** (see variables created above).

   The Command property is defined as **"select syaddressid from syaddress where systudentid = 51850"** where the systudentid value is hard-coded. Use a variable for the systudentid as appropriate in your environment.



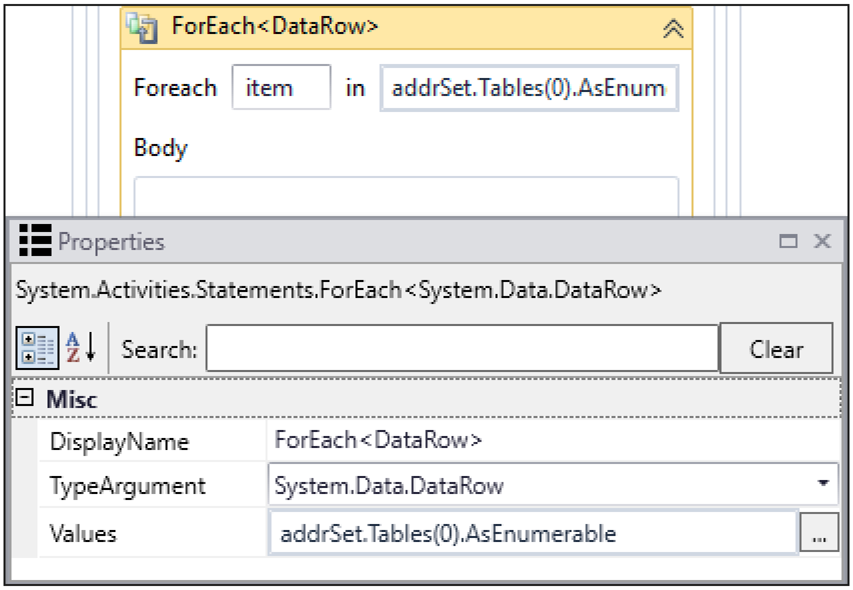
Connection string name

Command "select syaddressid from syaddress \

Cmc.Core.Workflow.Activities.ExecuteQuery

CommandText	"select syaddressid from syaddress where systudentid = 51850"
ConnectionStringName	*Enter a VB expression*
Data	addrSet
DisplayName	ExecuteQuery

8. Drag a ForEach<> activity below the ExecuteQuery activity. The ForEach<> is activity converts the dataset type argument returned by ExecuteQuery to a collection of Int32 ids to pass to GetEntityCollection<> activity using the Values property **addrSet.Tables(0).AsEnumerable**.



Foreach item in addrSet.Tables(0).AsEnum

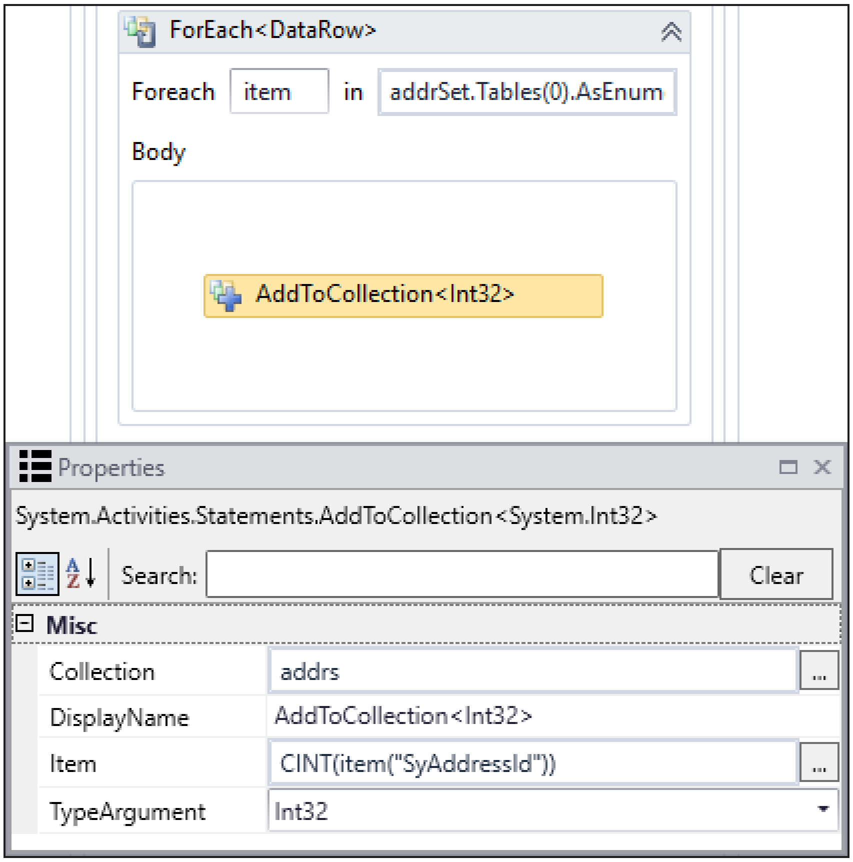
System.Activities.Statements.ForEach<System.Data.DataRow>

DisplayName	ForEach<DataRow>
TypeArgument	System.Data.DataRow
Values	addrSet.Tables(0).AsEnumerable

9. Drag an AddToCollection<> activity into the Body section of the ForEach<> activity. The AddToCollection activity adds items to the collection when users enter new data on the form.

   The collection is defined by the variable **addrs** of type **List<Int32>** with a default value of **new List(Of Int32)**.

   The **Item** property value **CINT(item("SyAddressId"))** converts the data to integers.

   Foreach item in addrSet.Tables(0).AsEnum

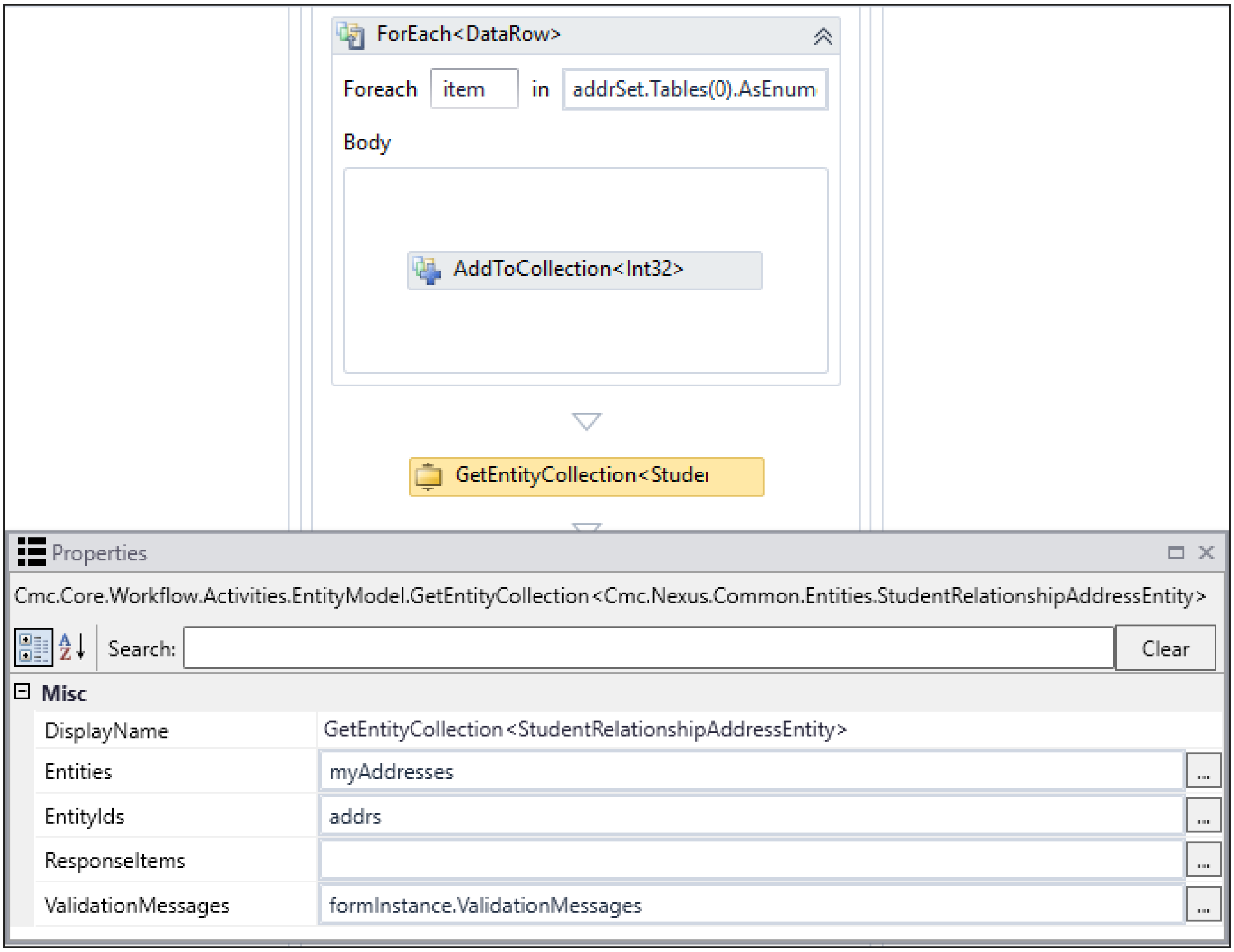
   System.Activities.Statements.AddToCollection<System.Int32>

Collection	addrs
DisplayName	AddToCollection<Int32>
Item	CINT(item("SyAddressId"))
TypeArgument	Int32

10. Drag **a GetEntityCollection<>** activity below the ForEach<> activity. The GetEntityCollection<> activity uses the **StudentRelationshipAddressEntity**.

    The input argument is the **addrs** variable.

    The output argument is the **myAddresses** argument that binds the grid to the workflow.



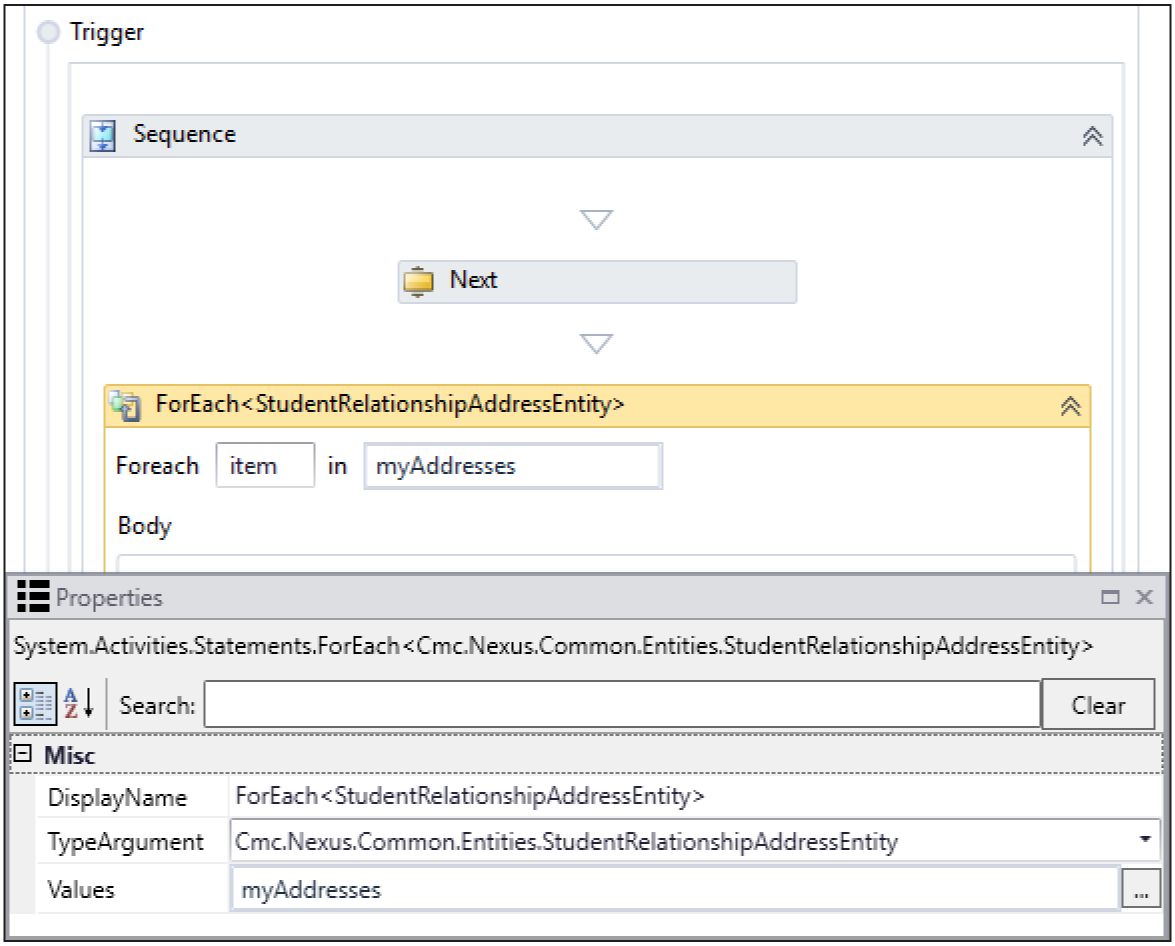
Cmc.Core.Workflow.Activities.EntityModel.GetEntityCollection<Cmc.Nexus.Common.Entities.StudentRelationshipAddressEntity>

DisplayName	GetEntityCollection<StudentRelationshipAddressEntity>
Entities	myAddresses
EntityIds	addrs
ResponseItems	
ValidationMessages	formInstance.ValidationMessages

11. Drag a ForEach<> activity into the Next transition following the form the contains the Grid component.

    The Values property holds the **myAddresses** argument that binds the grid to the workflow.

    This instance of the ForEach activity gathers all rows in the grid including rows that were added by the form user.



Foreach item in myAddresses

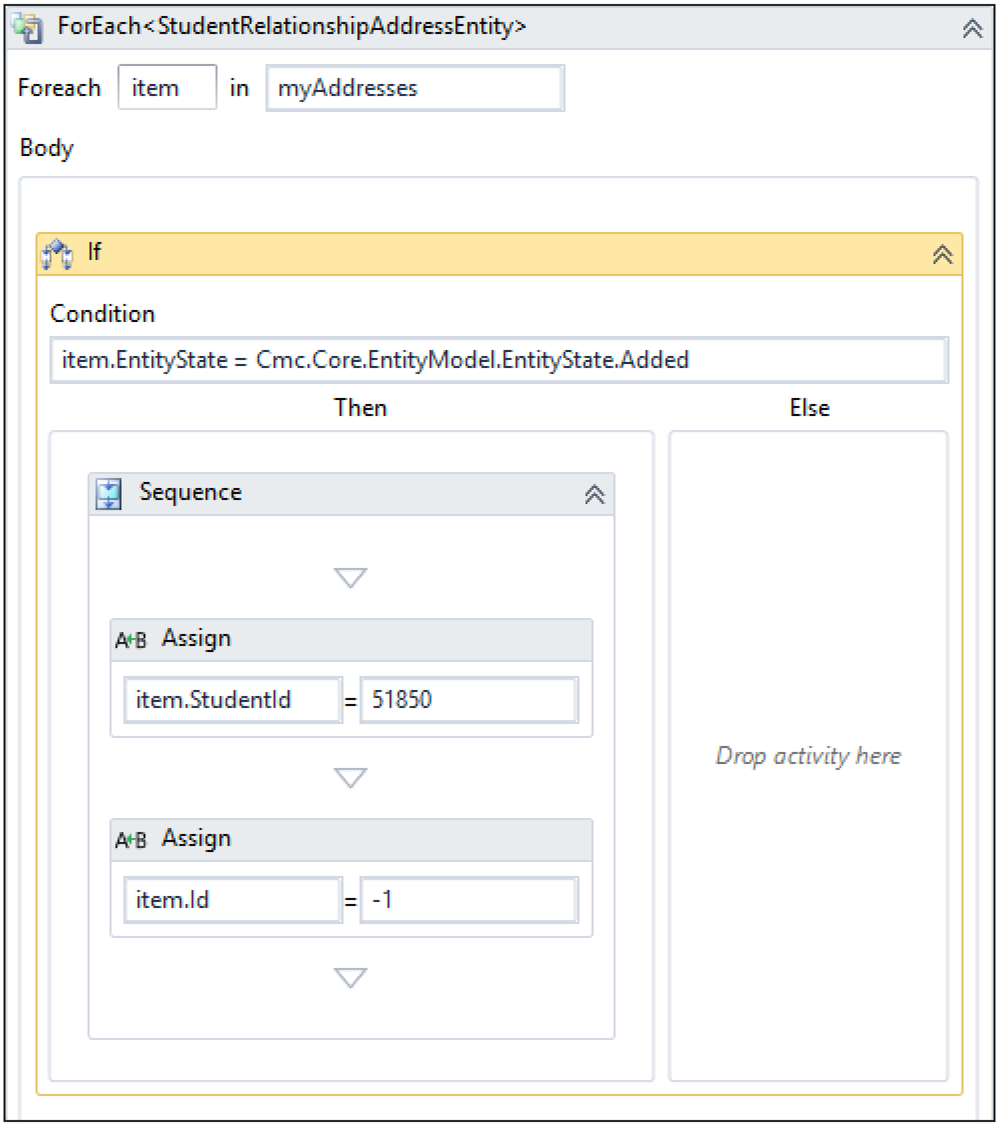
System.Activities.Statements.ForEach<Cmc.Nexus.Common.Entities.StudentRelationshipAddressEntity>

DisplayName	ForEach<StudentRelationshipAddressEntity>
TypeArgument	Cmc.Nexus.Common.Entities.StudentRelationshipAddressEntity
Values	myAddresses

12. Drag an **If** activity into the Body section of the ForEach<> activity. Specify the following condition to detect if an item was added to the StudentRelationshipAddressEntity:
item.EntityState = Cmc.Core.EntityModel.EntityState.Added

    Drag an Assign activity into the **Then** branch to the associate the hard-coded studentid with the itemEntityState array.

    Add another Assign activity to set the **item.Id** to **-1**. This assign statement ensures that a new item is appended to the array. The last element of an array is the length of the array - 1.



ForEach<StudentRelationshipAddressEntity>

Foreach item in myAddresses

Body

If

Condition

item.EntityState = Cmc.Core.EntityModel.EntityState.Added

Then

Sequence

Assign

item.StudentId = 51850

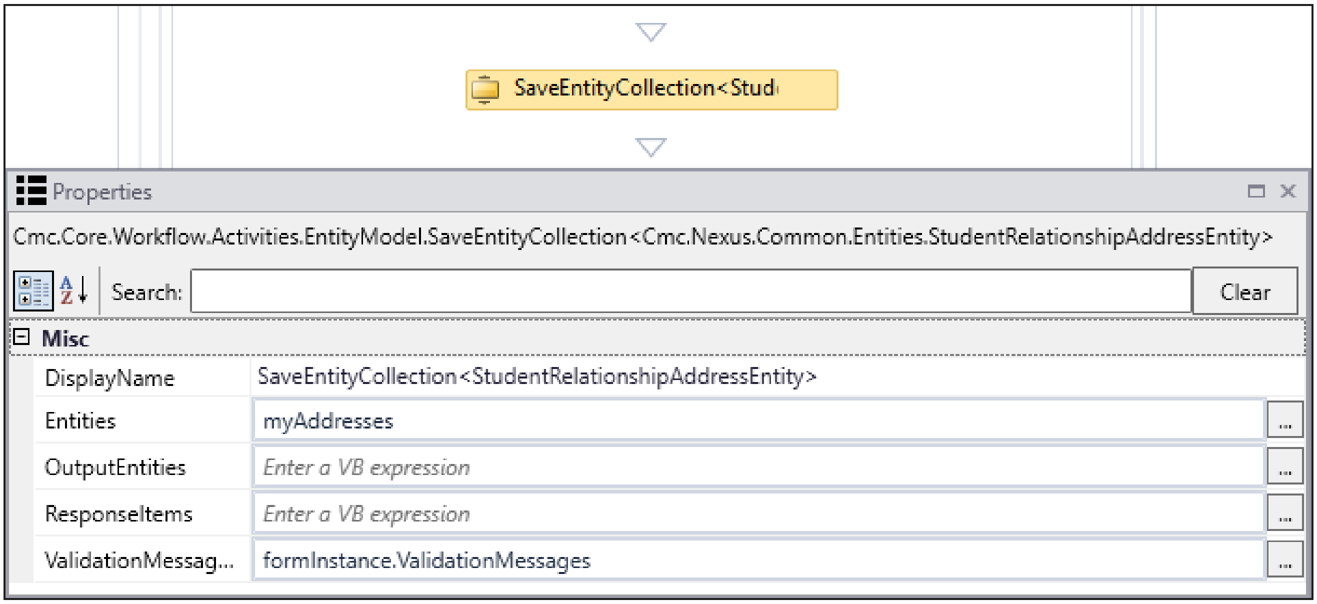
Assign

item.Id = -1

Else

Drop activity here

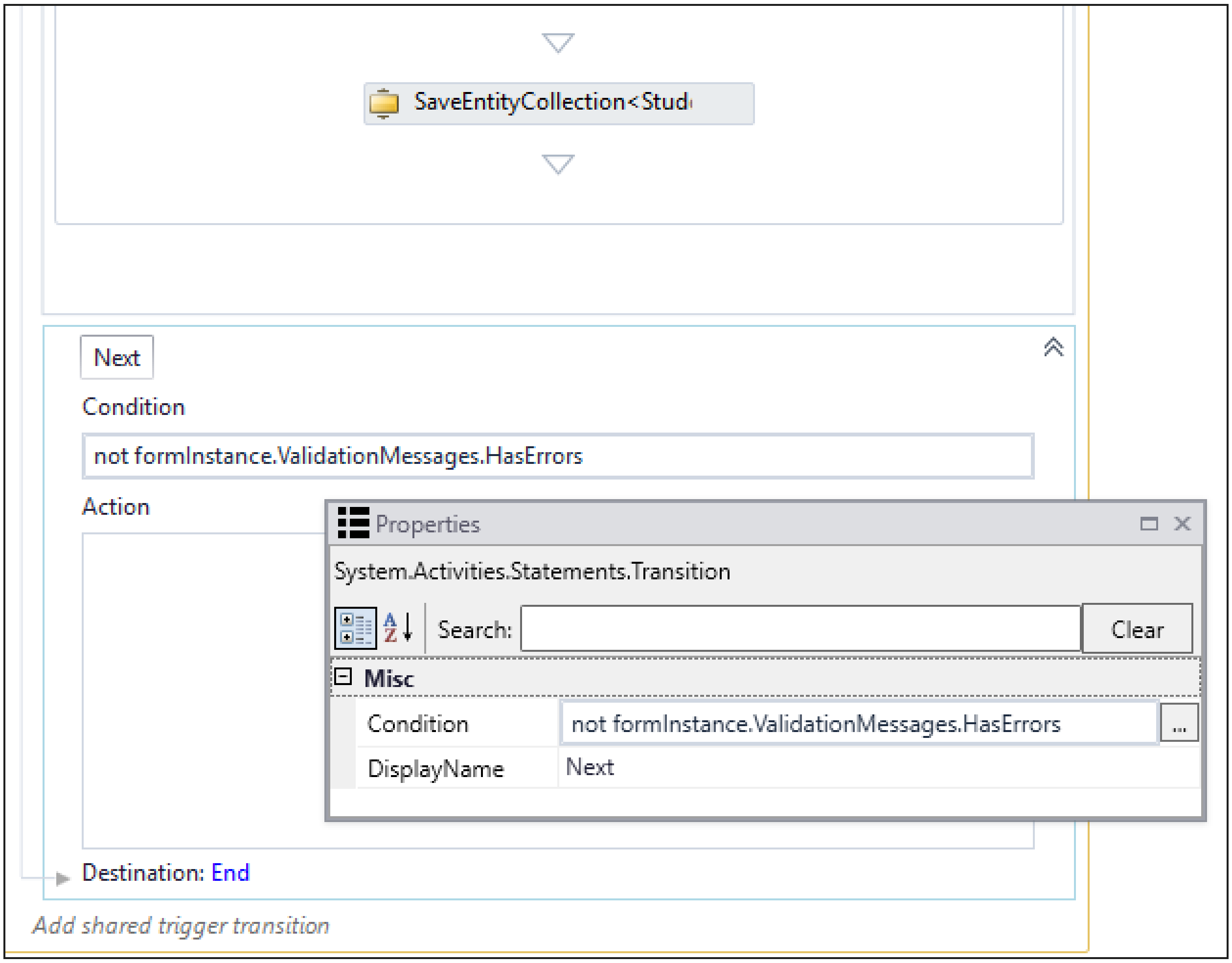
13. Drag a SaveEntityCollection<> activity into last Next transition of the sequence. The activity will handle add, edit and delete of any entity in the. In our example, the activity saves the changes passed in through **myAddresses** to the ICollection<StudentRelationshipAddressEntity>.



Cmc.Core.Workflow.Activities.EntityModel.SaveEntityCollection<Cmc.Nexus.Common.Entities.StudentRelationshipAddressEntity>

DisplayName	SaveEntityCollection<StudentRelationshipAddressEntity>
Entities	myAddresses
OutputEntities	*Enter a VB expression*
ResponseItems	*Enter a VB expression*
ValidationMessag...	formInstance.ValidationMessages

14. Finally, in the Condition field of the last Next transition, specify **not formInstance.ValidationMessages.HasErrors** to catch any form errors.



Condition

not formInstance.ValidationMessages.HasErrors

System.Activities.Statements.Transition

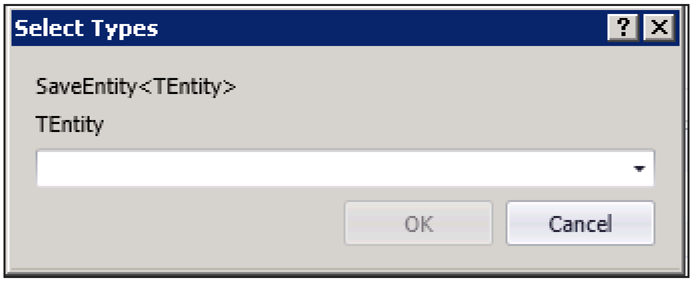
Condition	not formInstance.ValidationMessages.HasErrors
DisplayName	Next

Destination: End

SaveEntity<>

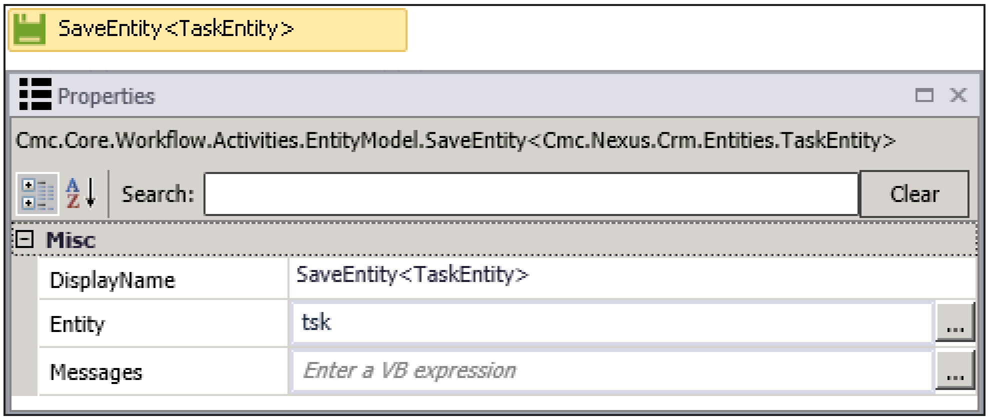
The SaveEntity<> activity uses an entity service to save an instance of an entity that was updated or created using a CreateEntity<> activity.

When you drag the SaveEntity<> activity into the Designer window, you are prompted to select the entity type.



When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.

After you have selected an entity, the name of the entity is inserted into the DisplayName field, e.g., SaveEntity<TaskEntity>. Proceed to specify the entity to be saved and, optionally, a validation message.



SaveEntity<TaskEntity>

Properties

Cmc.Core.Workflow.Activities.EntityModel.SaveEntity<Cmc.Nexus.Crm.Entities.TaskEntity>

Search:

Misc

DisplayName	SaveEntity<TaskEntity>
Entity	tsk
Messages	Enter a VB expression

Properties

SaveEntity<> Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Entity	`InOutArgument<Entity>`	Yes	Specify the entity previously created with a CreateEntity<> activity using a VB expression or variable.

Property	Value	Required	Notes
Messages	`InArgument <ICollection <ValidationMessage>>`	No	Specify the validation message to be displayed when the entity is saved. Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

The following properties are mandatory to save the following CampusNexus CRM entities:

Mandatory Properties for CRM Entities

CRM Entity	Mandatory Property
Contact	One of the following properties: • Name • First Name • Middle Name • Last Name
Lead	Team ID One of the following properties: • Name • First Name • Middle Name • Last Name
Account	Account name
Custom Object (Global)	Name
Custom Object (Team)	Name Team ID
Custom Object (Shared)	Name Shared to

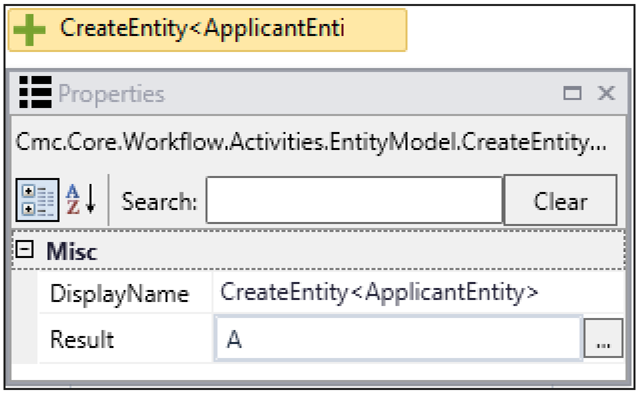
If you are working with the ApplicantEntity in Anthology Student, refer to Create/Save ApplicantEntity and Update Derived Fields.

Create/Save ApplicantEntity and Update Derived Fields

When the CreateEntity<> and SaveEntity<> activities are used with the ApplicantEntity in Anthology Student, the workflow must include a number of Assign activities to ensure that a record is added to the AdEnroll table the same way as when the Anthology Student client is used to add an applicant record. In addition, it is necessary to retrieve the IApplicantServerice event using the GetServiceInstance<> activity (steps 6 and 7) to update the derived fields.

The following steps are required in the workflow:

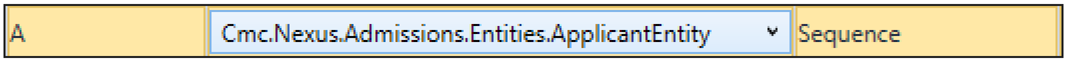
1. Drag a **CreateEntity<ApplicantEntity>**activity into the sequence to create an instance of the ApplicantEntity record.

   Cmc.Core.Workflow.Activities.EntityModel.CreateEntity...

DisplayName	CreateEntity<ApplicantEntity>
Result	A

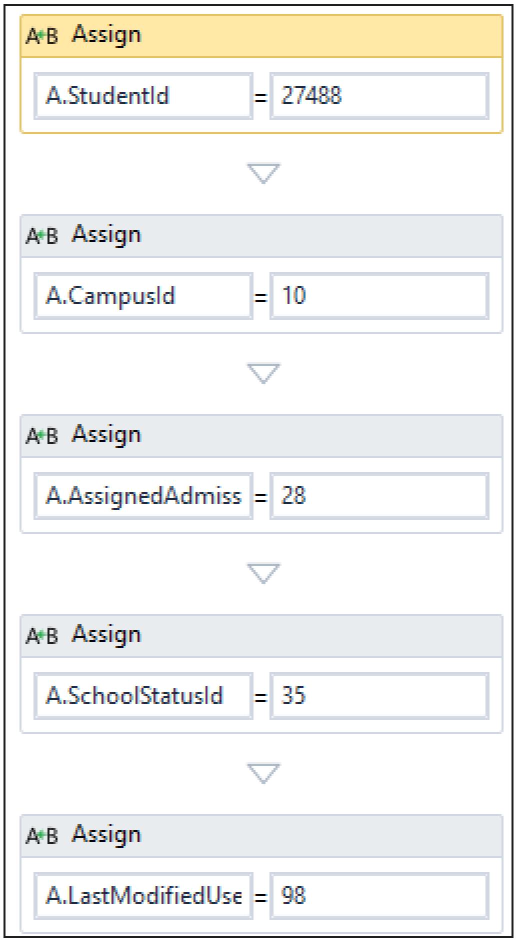
   The Result value "A" is defined as a variable with the following attributes. This variable is referenced in the Assign activities below.

A	Cmc.Nexus.Admissions.Entities.ApplicantEntity	Sequence

2. Use **Assign** activities to populate the following **required** fields:

   - **StudentId**
   - **CampusId**
   - **AssignedAdmissionsRepId**
   - **SchoolStatusId**
   - **LastModifiedUserId**



Assign

A.StudentId = 27488

Assign

A.CampusId = 10

Assign

A.AssignedAdmiss = 28

Assign

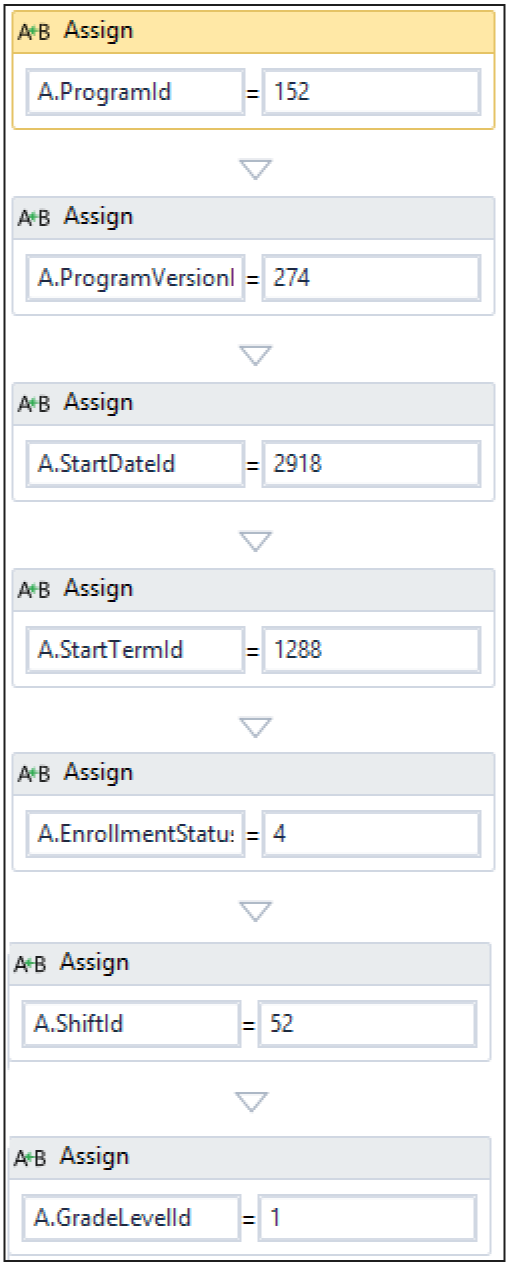
A.SchoolStatusId = 35

Assign

A.LastModifiedUse = 98

3. Use **Assign** activities to populate the following **optional** fields:

   - **ProgramId**
   - **ProgramVersionId**
   - **StartDateId**
   - **StartTermId**
   - **EnrollmentStatusId**
   - **ShiftId**
   - **GradeLevelId**



Assign
A.ProgramId = 152

Assign
A.ProgramVersionI = 274

Assign
A.StartDateId = 2918

Assign
A.StartTermId = 1288

Assign
A.EnrollmentStatu: = 4

Assign
A.ShiftId = 52

Assign
A.GradeLevelId = 1

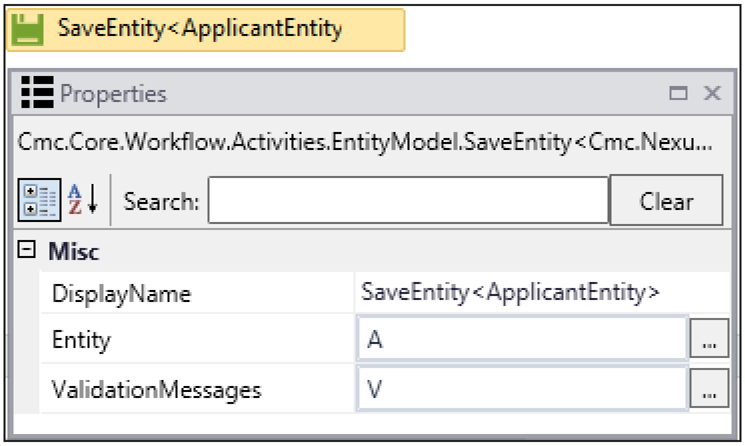
With these assignments the workflow calculates values for the remaining ApplicantEntity fields and passes the values to the SaveEntity<ApplicantEntity> activity.

The values are calculated (derived) for the following fields:

- BillingMethodId
- GradeScaleId
- ClockHoursRequired
- CreditHoursRequired
- ExpectedStartDate
- MidpointDate
- GraduationDate
- CatalogYearId

Note: If the Program Version has more than one Catalog, the logic identifies the CatalogYearID whose Start Date is nearest to the selected Program Version Start Date.

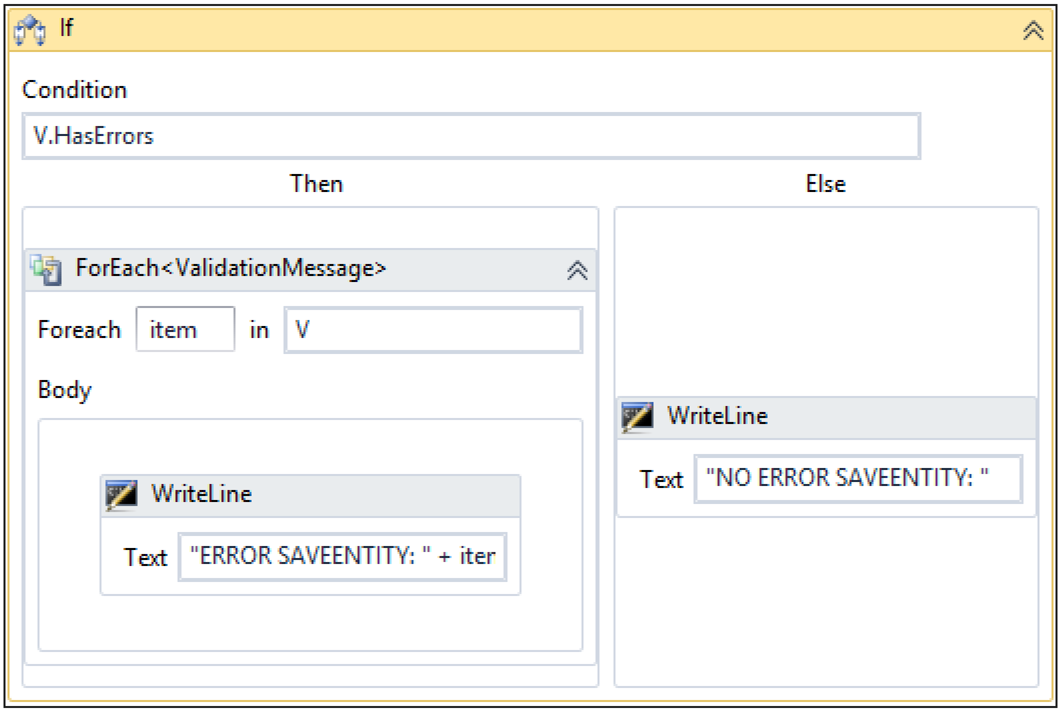
4. Insert a **SaveEntity<ApplicantEntity>** activity into the workflow. This activity populates the entire applicant record in the AdEnroll table using the assigned and calculated (derived) values.



Cmc.Core.Workflow.Activities.EntityModel.SaveEntity<Cmc.Nexu...

DisplayName	SaveEntity<ApplicantEntity>
Entity	A
ValidationMessages	V

5. Optionally, add an If condition to the workflow to catch validation errors. In our example "V" is a variable of type Cmc.Core.EventingValidationMessageCollection.



Condition: V.HasErrors

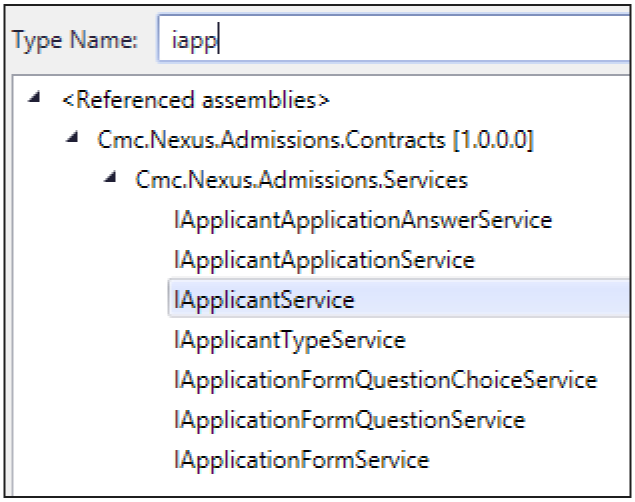
Foreach item in V

Text "ERROR SAVEENTITY: " + iter

Text "NO ERROR SAVEENTITY: "

6. Drag a **GetServiceInstance** activity into the workflow. The Select Types dialog is displayed.

Click Browse for Types, select **IApplicantService**, and click OK.

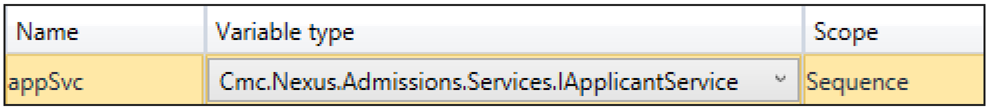


Type Name: iapp

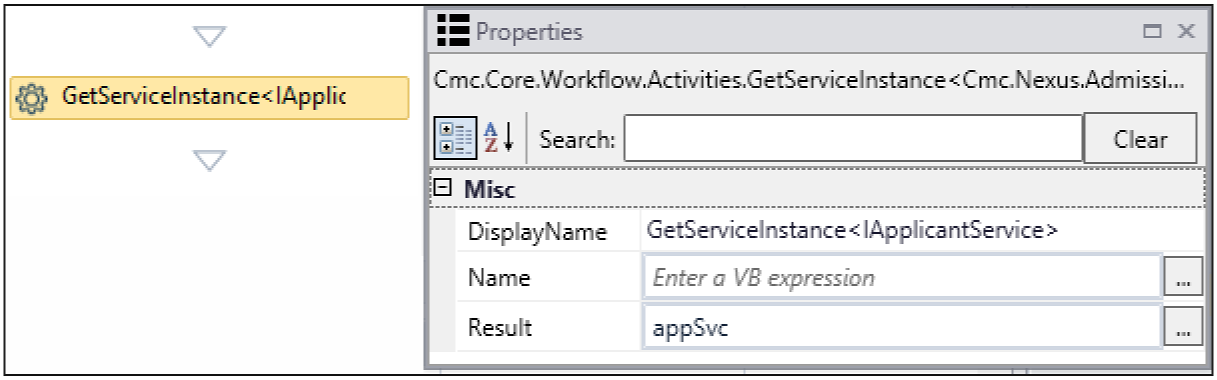
- <Referenced assemblies>
  - Cmc.Nexus.Admissions.Contracts [1.0.0.0]
    - Cmc.Nexus.Admissions.Services
      - IApplicantApplicationAnswerService
      - IApplicantApplicationService
      - IApplicantService
      - IApplicantTypeService
      - IApplicationFormQuestionChoiceService
      - IApplicationFormQuestionService
      - IApplicationFormService

The Display Name of the GetServiceInstance activity is updated to GetServiceInstance<IApplicantService>.

In the Result property, specify a variable of type **Cmc.Nexus.Admissions.ServiceIApplicantService**.



Name	Variable type	Scope
appSvc	Cmc.Nexus.Admissions.Services.IApplicantService	Sequence



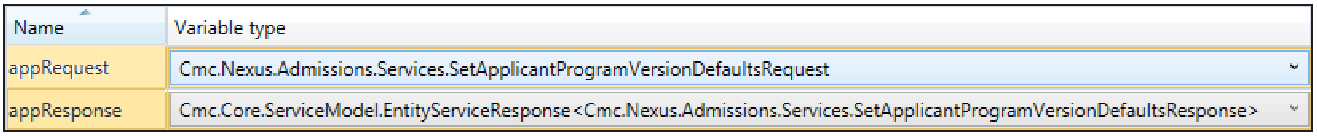
Cmc.Core.Workflow.Activities.GetServiceInstance<Cmc.Nexus.Admissi...

Search: Clear

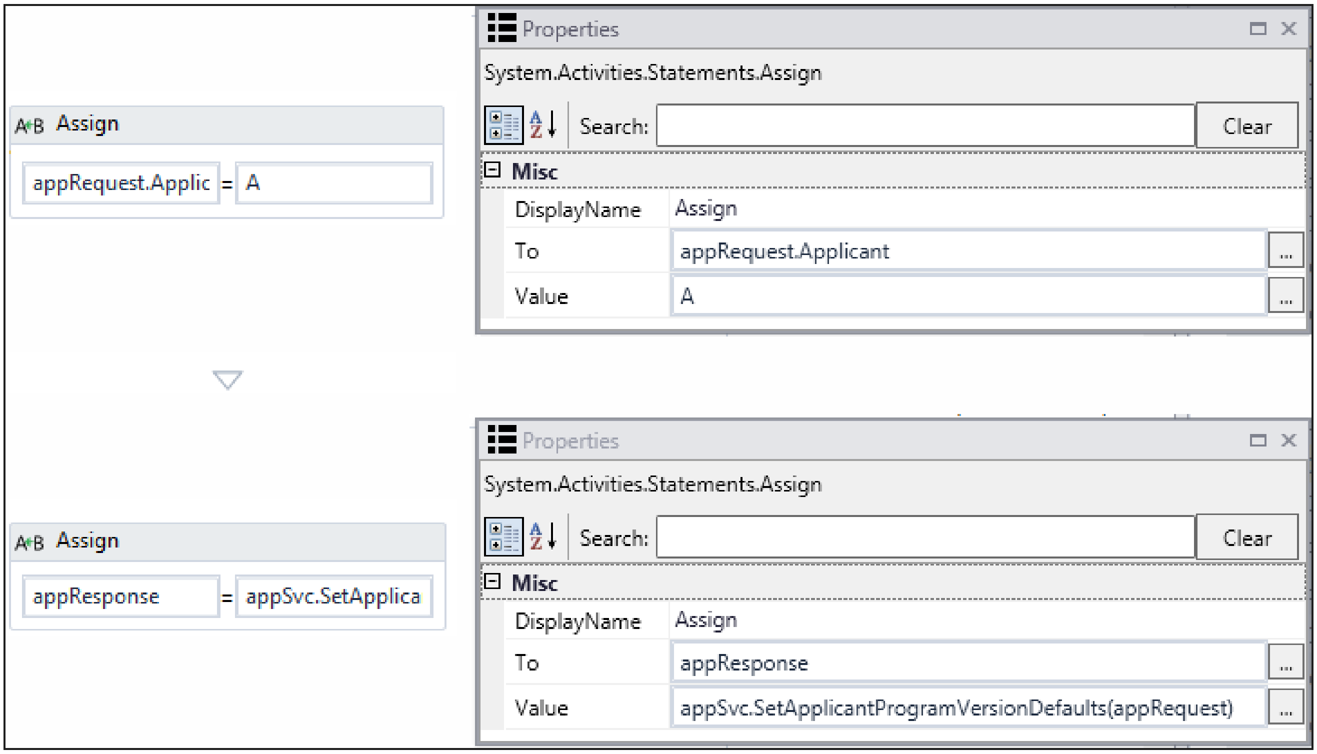
Misc

DisplayName	GetServiceInstance<IApplicantService>
Name	Enter a VB expression
Result	appSvc

7. Drag two **Assign** activities below the GetServiceInstance activity. Assign request and response variables for the applicant service defaults.



Name	Variable type
appRequest	Cmc.Nexus.Admissions.Services.SetApplicantProgramVersionDefaultsRequest
appResponse	Cmc.Core.ServiceModel.EntityServiceResponse<Cmc.Nexus.Admissions.Services.SetApplicantProgramVersionDefaultsResponse>



Notes:

- When an Applicant record is added and the **Note** field is **not** assigned, the Status History Comment text box in Anthology Student displays the hardcoded string *"Added Applicant record"*.
- When an Applicant record is added and the **Note** field is assigned in the workflow, the Status History Comment text box in Anthology Student displays the text of **entity.Note** that was passed into the workflow.

Create/Save StudentEntity

When the CreateEntity<> and SaveEntity<> activities are used with the StudentEntity in Anthology Student, the `IsActive` value in the SyStudent table is automatically set to `true` (1). This allows further activities on the student record to be executed, for example, Create/Save StudentPreviousEducationEntity.

SaveEntityCollection<>

Prerequisites

The SaveEntityCollection<> activity is available in Workflow Composer version 2.7 and later and requires the following **minimum** versions of activities and contracts:

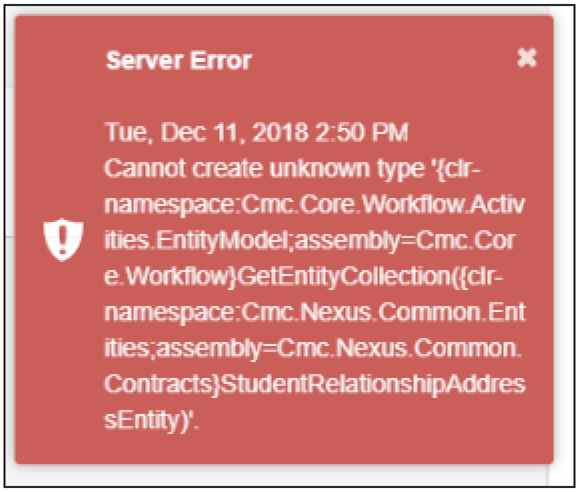
- Anthology Student version 20.0.x

  — OR —

- CampusNexus CRM version 12.2.x

The minimum Cmc.Core.dll version installed in Program Files (x86)\CMC\Workflow must be 5.1.167 or greater.

Note: If you use the activity with Student 19.0 and Workflow Composer 2.7, you won't see any errors in Workflow Composer (because it has minimum Cmc.Core.dll version), but you'll see a server error at runtime.



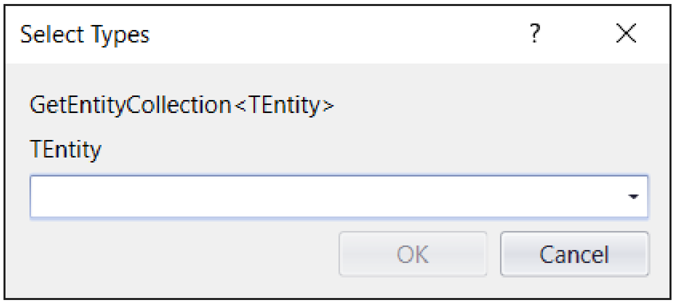
Server Error

Tue, Dec 11, 2018 2:50 PM
Cannot create unknown type '{clr-namespace:Cmc.Core.Workflow.Activities.EntityModel;assembly=Cmc.Core.Workflow}GetEntityCollection({clr-namespace:Cmc.Nexus.Common.Entities;assembly=Cmc.Nexus.Common.Contracts}StudentRelationshipAddressEntity)'.

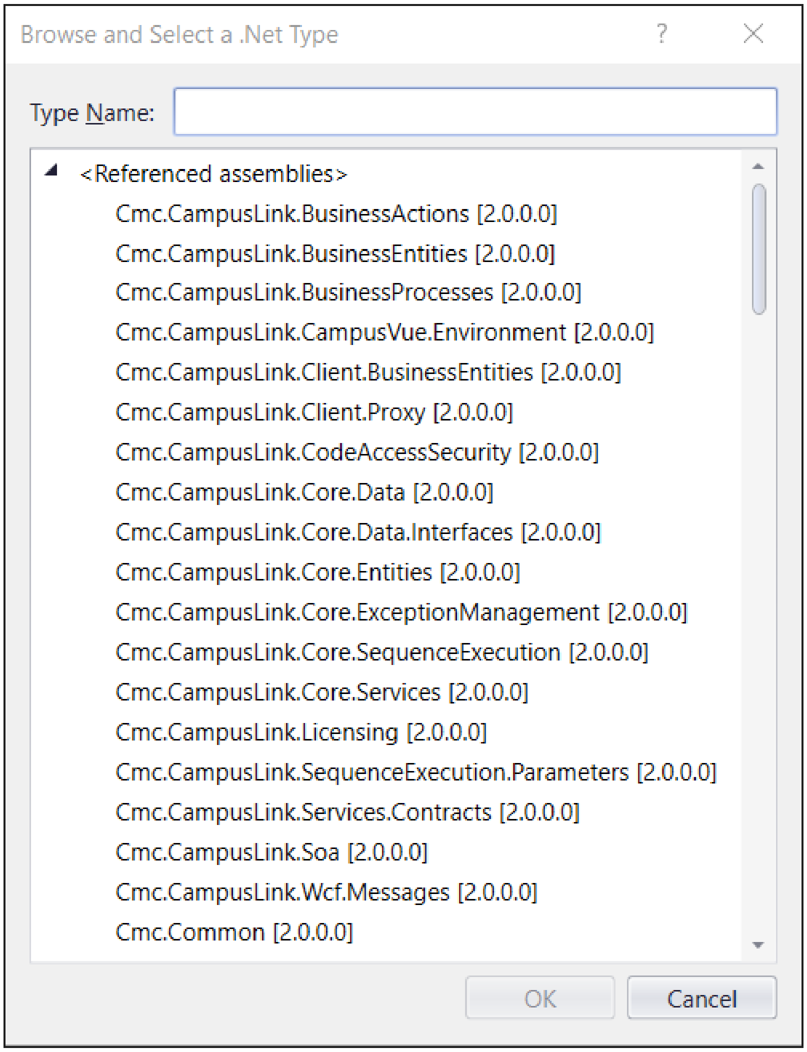
Purpose

The SaveEntityCollection<> provides the ability to pass in an entity collection retrieved using the GetEntityCollection<> activity and save the data for each instance of the collection.

When you drag the GetEntityCollection<> activity into the Designer window, you are prompted to select the entity type (TEntity).



When you select the 'Browse for Type' option, the list of assemblies and associated entities is displayed. Find and select the entity and click **OK**.



Browse and Select a .Net Type

Type Name:

<Referenced assemblies>
- Cmc.CampusLink.BusinessActions [2.0.0.0]
- Cmc.CampusLink.BusinessEntities [2.0.0.0]
- Cmc.CampusLink.BusinessProcesses [2.0.0.0]
- Cmc.CampusLink.CampusVue.Environment [2.0.0.0]
- Cmc.CampusLink.Client.BusinessEntities [2.0.0.0]
- Cmc.CampusLink.Client.Proxy [2.0.0.0]
- Cmc.CampusLink.CodeAccessSecurity [2.0.0.0]
- Cmc.CampusLink.Core.Data [2.0.0.0]
- Cmc.CampusLink.Core.Data.Interfaces [2.0.0.0]
- Cmc.CampusLink.Core.Entities [2.0.0.0]
- Cmc.CampusLink.Core.ExceptionManagement [2.0.0.0]
- Cmc.CampusLink.Core.SequenceExecution [2.0.0.0]
- Cmc.CampusLink.Core.Services [2.0.0.0]
- Cmc.CampusLink.Licensing [2.0.0.0]
- Cmc.CampusLink.SequenceExecution.Parameters [2.0.0.0]
- Cmc.CampusLink.Services.Contracts [2.0.0.0]
- Cmc.CampusLink.Soa [2.0.0.0]
- Cmc.CampusLink.Wcf.Messages [2.0.0.0]
- Cmc.Common [2.0.0.0]

Note that the in and out arguments for the activity are of type **ICollection**.

Properties

SaveEntityCollection<> Properties

Property	Value	Required	Notes
DisplayName	`String`	No	Specify a name for the activity or accept the default.
Entities	`InArgument<ICollection<Entity>>`	Yes	Specify the input entity array using a VB expression or variable.

Property	Value	Required	Notes
OutputEntities	`OutArgument<Icollection<Entity>>`	No	Specify the output entity array using a VB expression or variable.
ResponseItems	`OutArgument<ICollection <EntityServiceResponse<Entity>>>`	No	The items saved by this workflow activity. This is a variable that can be used as input for subsequent workflow activities. To identify the variable type, in the Variable type field of the Variables pane, select **Browse for Types...**. In the 'Browse and Select a .NET Type' window, navigate to the entity that matches the previously selected entity type and click **OK**.
ValidationMessages	`InArgument <ICollection<ValidationMessage>>`	No	Specify a variable that can be used to capture validation messages. For more information, see Capture Validation Errors.

For more information, see Get/Save EntityCollection Example.

Events in the New Object Model

The Anthology object model defines a collection of classes and interfaces through which entities can be manipulated. An entity represents a person, place, or thing such as a course, task, or campaign. Entities only contain the properties associated with itself such as first name, last name, or city. The verbs associated with an entity (e.g. Save, PostFinalGrades, or AddToCampaign) are exposed by a corresponding service or EntityService.

EntityModel

All entities in Anthology inherit from the Cmc.Core.EntityModel.Entity abstract base class. The Entity base class contains all the logic required for maintaining the state of an entity and its children while it is being modified in business logic, on the client, or by an external system. Each entity is defined through its properties and the methods it supports. The exposed (public) properties and methods of an entity can be manipulated through workflows.

The following are examples of properties and methods that can be associated with an entity.

Properties

- EntityState — gets or sets the state of an entity
    - Added — the entity is new, an INSERT database operation will be performed
    - Removed — the entity has been removed, a DELETE operation will be performed
    - Modified — the entity has been modified, an UPDATE database operation will be performed
    - Unchanged — the entity is unchanged, no database operation will be performed
- ExtendedProperties — represents a collection of dynamic entity properties such as School Defined Fields.
- ModifiedProperties — represents a read-only collection of property names that have been modified since the entity was last retrieved.
- OriginalState — represents the entity’s original state serialized in a byte[]. This property is used to round-trip the entity state from the client to the server and is not intended to be updated directly in code.
- OriginalValues — represents the original values of an entity as a dictionary.

Methods

- AcceptChanges — accepts all current changes and sets the entity’s state to Unchanged. This does not perform a database operation.
- GetOriginalValue — gets the original value of a specified property
- HasChanged — returns true if an entity (or its children) have changed; else false

Events Raised by EntityState Changes

The EntityState property is exposed in many Anthology entities. This property supports create, retrieve, update, delete (CRUD) operations or commands. The create, update, and delete operations raise events associated with the affected entities. Workflows can be triggered by any create, update, or delete operation. Retrieve or get operations do not trigger events.

The CreateEntity<>, DeleteEntity<>, GetEntity<>, and SaveEntity<> activities in Workflow Composer under Cmc.Core.Workflow.Activities.EntityModel enable you to access the EntityState property of exposed entities and to manipulate and persist the state of an entity in the database.

Event Handlers

The object model provides event handlers for all entities. The event handlers support event services for each entity.

The Event Broker listens for incoming events from clients, determines the name of the event, forwards the event to the configured event handler, and, if required, returns a response to the event.

Event messages contain enough basic information to be handled without the need to retrieve additional data from APIs.

The events that are exposed to the Event Broker can be consumed in custom code (for example, C# event handlers) or workflows that automate tasks and enable data to be exchanged between systems.

Anthology events are grouped in the following categories:

- Constructed events enable new objects to be added to the database.
- Deleting or Deleted events enable objects to be deleted.
- Saving or Saved events enable objects to be inserted/updated.

Saving events and Deleting events are captured and visible at the UI level. VB .NET code is required to intercept these events. Typically, data validation occurs typically occurs. Saving and Deleting event workflows must be stored on the host that is running the application on which the event is captured, for example, Anthology Student.

Saved events and Deleted events are captured at the database trigger level when a transaction is committed to the database. These events are only visible in the event log of the Windows Service NextGen Nexus Event Workflows. Saved and Deleted event workflows must be stored on a host that has a direct database connection, for example, COM Server.

Constructed events enable new objects to be added. Constructed events are captured and visible at the UI level when the components of a record are assembled. No data validation occurs. VB .NET code is required to intercept these events.

EntityServices

The Anthology object model contains numerous entity services such as the Cmc.Nexus.Academics.Services that support custom commands. The Cmc.Nexus.Academics.Services, for example, raise the Cmc.Nexus.Academics.Events which contain custom commands related to the Academics module. The event handlers of the entities contain business logic applicable to the entities. The event handlers can be extended using workflows, for example workflows that send emails to advisors when a student unregisters from a class.

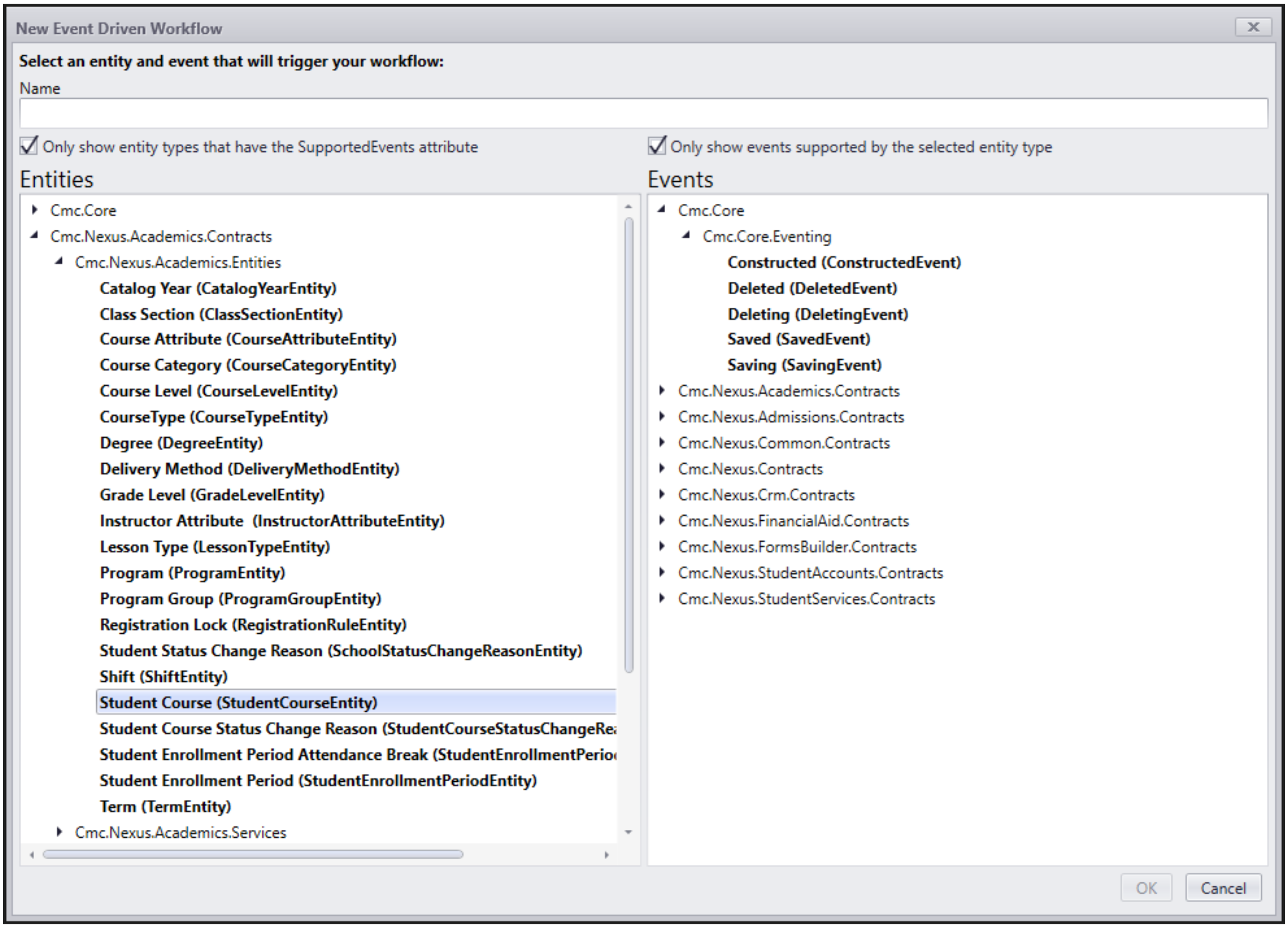
Selecting Events in Workflow Composer

The 'New Event Driven Workflow' window in Workflow Composer enables you to select the events that will trigger your workflow.

The Entities filter option **Only show entity types that have the SupportedEvents attribute** is selected by default. The SupportedEvents attribute indicates what type of events are supported by an entity or service. When any one of the supported events is enabled, the entity is visible to Workflow Composer and considered public.

The Events filter option **Only show events supported by the selected entity type** is also selected by default and makes it easier to find supported events after selecting an Entity.

The Entities pane below shows the entities in the Cmc.Nexus.Academics.Entities namespace. The Events pane shows the events that are available (Constructed, Deleted, Deleting, Saved, Saving events) when Student Course (StudentCourseEntity) is selected in the Cmc.Nexus.Academics.Entities namespace.



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

Entities

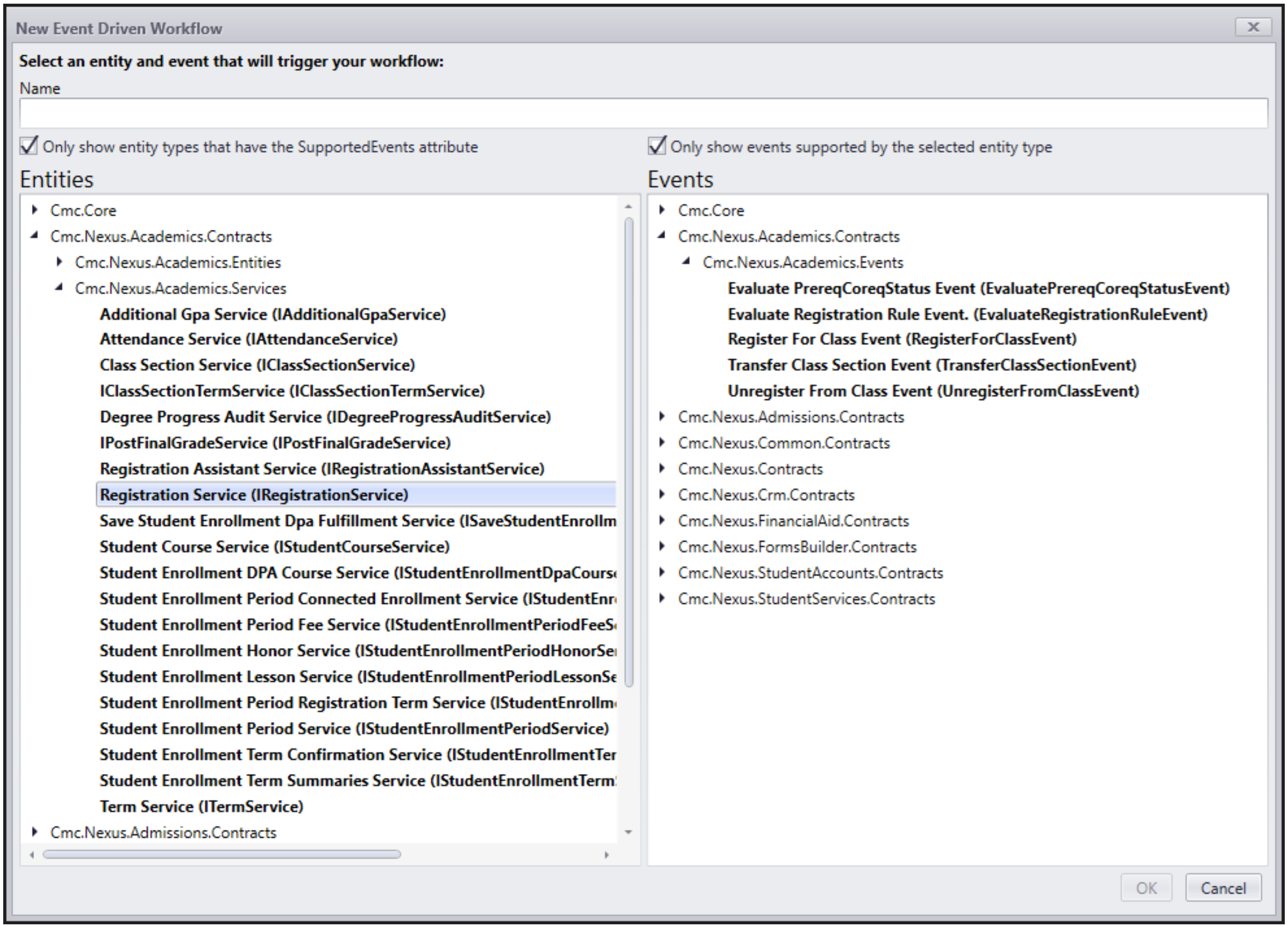
- Cmc.Core
- Cmc.Nexus.Academics.Contracts
  - Cmc.Nexus.Academics.Entities
    - **Catalog Year (CatalogYearEntity)**
    - **Class Section (ClassSectionEntity)**
    - **Course Attribute (CourseAttributeEntity)**
    - **Course Category (CourseCategoryEntity)**
    - **Course Level (CourseLevelEntity)**
    - **CourseType (CourseTypeEntity)**
    - **Degree (DegreeEntity)**
    - **Delivery Method (DeliveryMethodEntity)**
    - **Grade Level (GradeLevelEntity)**
    - **Instructor Attribute (InstructorAttributeEntity)**
    - **Lesson Type (LessonTypeEntity)**
    - **Program (ProgramEntity)**
    - **Program Group (ProgramGroupEntity)**
    - **Registration Lock (RegistrationRuleEntity)**
    - **Student Status Change Reason (SchoolStatusChangeReasonEntity)**
    - **Shift (ShiftEntity)**
    - **Student Course (StudentCourseEntity)**
    - **Student Course Status Change Reason (StudentCourseStatusChangeRe**
    - **Student Enrollment Period Attendance Break (StudentEnrollmentPerio**
    - **Student Enrollment Period (StudentEnrollmentPeriodEntity)**
    - **Term (TermEntity)**
  - Cmc.Nexus.Academics.Services

Events

- Cmc.Core
  - Cmc.Core.Eventing
    - **Constructed (ConstructedEvent)**
    - **Deleted (DeletedEvent)**
    - **Deleting (DeletingEvent)**
    - **Saved (SavedEvent)**
    - **Saving (SavingEvent)**
- Cmc.Nexus.Academics.Contracts
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.Common.Contracts
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.FormsBuilder.Contracts
- Cmc.Nexus.StudentAccounts.Contracts
- Cmc.Nexus.StudentServices.Contracts

EntityServices are typically associated with custom events and business rules that apply to an entity. The events are associated with an EntityService are displayed in the Events pane when you select a service in the Entities pane.

The Entities pane below shows the services in the Cmc.Nexus.Academics.Entities namespace. The Events pane shows the events that are available (Register for Class Event, Transfer Class Section Event, and Unregister From Class Event) when Registration Service (IRegistrationService) is selected in the Cmc.Nexus.Academics.Entities namespace.



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

☑ Only show events supported by the selected entity type

Entities

- Cmc.Core
- Cmc.Nexus.Academics.Contracts
  - Cmc.Nexus.Academics.Entities
  - Cmc.Nexus.Academics.Services
    - **Additional Gpa Service (IAdditionalGpaService)**
    - **Attendance Service (IAttendanceService)**
    - **Class Section Service (IClassSectionService)**
    - **IClassSectionTermService (IClassSectionTermService)**
    - **Degree Progress Audit Service (IDegreeProgressAuditService)**
    - **IPostFinalGradeService (IPostFinalGradeService)**
    - **Registration Assistant Service (IRegistrationAssistantService)**
    - **Registration Service (IRegistrationService)**
    - **Save Student Enrollment Dpa Fulfillment Service (ISaveStudentEnrollm**
    - **Student Course Service (IStudentCourseService)**
    - **Student Enrollment DPA Course Service (IStudentEnrollmentDpaCours**
    - **Student Enrollment Period Connected Enrollment Service (IStudentEnr**
    - **Student Enrollment Period Fee Service (IStudentEnrollmentPeriodFeeS**
    - **Student Enrollment Honor Service (IStudentEnrollmentPeriodHonorSe**
    - **Student Enrollment Lesson Service (IStudentEnrollmentPeriodLessonSe**
    - **Student Enrollment Period Registration Term Service (IStudentEnrollm**
    - **Student Enrollment Period Service (IStudentEnrollmentPeriodService)**
    - **Student Enrollment Term Confirmation Service (IStudentEnrollmentTer**
    - **Student Enrollment Term Summaries Service (IStudentEnrollmentTerm**
    - **Term Service (ITermService)**
- Cmc.Nexus.Admissions.Contracts

Events

- Cmc.Core
- Cmc.Nexus.Academics.Contracts
  - Cmc.Nexus.Academics.Events
    - **Evaluate PrereqCoreqStatus Event (EvaluatePrereqCoreqStatusEvent)**
    - **Evaluate Registration Rule Event. (EvaluateRegistrationRuleEvent)**
    - **Register For Class Event (RegisterForClassEvent)**
    - **Transfer Class Section Event (TransferClassSectionEvent)**
    - **Unregister From Class Event (UnregisterFromClassEvent)**
- Cmc.Nexus.Admissions.Contracts
- Cmc.Nexus.Common.Contracts
- Cmc.Nexus.Contracts
- Cmc.Nexus.Crm.Contracts
- Cmc.Nexus.FinancialAid.Contracts
- Cmc.Nexus.FormsBuilder.Contracts
- Cmc.Nexus.StudentAccounts.Contracts
- Cmc.Nexus.StudentServices.Contracts

The public Anthology Student entities and event services are documented in the Anthology Student Object Library. Use to the library to look up details about Anthology entities including classes, properties, event arguments, methods, and fields while building workflows.

Generic Activities

Workflow Designer is built using the Windows Workflow Foundation (WF) in the .NET Framework. It contains Microsoft's built-in (generic) workflow activities and activities created specifically for Anthology products (CMC Activities).

The Microsoft WF activity library contains the activities described below. These activities are used in conjunction with the CMC Activities developed for Anthology.

For detailed information about WF features first introduced in .NET 4.5 refer to http://msdn.microsoft.com/en-us/library/vstudio/hh305677(v=vs.110).aspx.

Collection

Collection activities are used to work with collection objects in a workflow. The .NET Framework has system-provided activities for adding and removing items from a collection, testing for the existence of an item in a collection, and clearing a collection. ExistsInCollection and RemoveFromCollection have an OutArgument of type Boolean, which indicates the result.

Collection Activities

Activity	Description
AddToCollection<>	Adds an item to a specified collection.
ClearCollection<>	Clears all items from a specified collection.
ExistsInCollection<>	Returns **true** if an item exists in a collection.
RemoveFromCollection<>	Removes an item from a specified collection and returns **true** if the item was successfully removed.

For more information, see http://msdn.microsoft.com/en-us/library/vstudio/ee358729(v=vs.100).aspx.

Control Flow

The .NET Framework provides several activities for controlling flow of execution within a workflow. Some of these activities (such as Switch and If) implement flow control structures similar to those in programming environments such as Visual C#, while others (such as Pick) model new programming structures.

Note that while activities such as the Parallel and ParallelForEach activities schedule multiple child activities for execution simultaneously, only a single thread is used for a workflow. Each child activity of these activities executes sequentially and successive activities do not execute until previous activities either complete or go idle. As a result, these activities are most useful for applications in which several potentially blocking activities must execute in an interleaved fashion. If none of the child activities of these activities go idle, a Parallel activity executes just like a Sequence activity, and a ParallelForEach activity executes just like a ForEach activity. If, however, asynchronous activities (such as activities that derive from AsyncCodeActivity) or messaging activities are used, control will pass to the next branch while the child activity waits for its message to be received or its asynchronous work to be completed.

Control Flow Activities

Activity	Description
DoWhile	Executes the contained activities once and continues to do so while a condition is true.
ForEach<>	Executes an embedded statement in sequence for each element in a collection. ForEach is similar to the keyword foreach but is implemented as an activity rather than a language statement.
If	Executes contained activities if a condition is true and can execute activities contained in the Else property if the condition is false.
Parallel	Executes contained activities in parallel.
ParallelForEach<>	Executes an embedded statement in parallel for each element in a collection.
Pick	Provides event-based control flow modeling.
PickBranch	Represents a potential path of execution in a Pick activity.
Sequence	Executes contained activities in sequence.
Switch<>	Selects one choice from a number of activities to execute, based on the value of a given expression.
While	Executes contained activities while a condition is true.

For more information about the classes, methods, and properties associated with each activity, refer to http://msdn.microsoft.com/en-us/library/vstudio/ee358737(v=vs.100).aspx.

Error Handling

The .NET Framework provides several system-provided activities for implementing error handling and recovery.

Error Handling Activities

Activity	Description
Rethrow	Rethrows the last exception thrown from within a TryCatch activity.
Throw	Throws an exception.
TryCatch	Implements exception handling.

For more information, see http://msdn.microsoft.com/en-us/library/vstudio/ee358726(v=vs.100).aspx.

State Machine

The .NET Framework provides several system-provided activities and activity designers for creating state machine workflows.

State Machine Activities

Activity	Description
FinalState	Represents a terminating state in a state machine. FinalState is an activity designer that when used creates a State preconfigured as a terminating state. For more information, see FinalState Activity Designer.
State	Represents a state in a state machine.
StateMachine	Executes contained activities using the familiar state machine paradigm.
Transition	Represents the transition between two states. There is no Toolbox item for Transition; transitions are created on the workflow designer by dragging and dropping a line between two states, or by dropping a state on the triangles that appear when one state is hovered over another.

For more information, see http://msdn.microsoft.com/en-us/library/vstudio/gg983475(v=vs.100).aspx.

Flowchart

The .NET Framework provides several system-provided activities for controlling execution and branching within a Flowchart.

Flowchart Activities

Activity	Description
Flowchart	Executes contained activities using the familiar Flowchart paradigm.
FlowDecision	A specialized FlowNode that provides the ability to model a conditional node with two outcomes.
FlowSwitch<>	A specialized FlowNode that allows modeling a switch construct, with one expression of a type defined in the activity's type specifier and a single outcome for each match.

For more information, see http://msdn.microsoft.com/en-us/library/vstudio/ee358753(v=vs.100).aspx.

Messaging

Messaging activities allow workflows to send and receive WCF messages. By adding messaging activities to a workflow you can model any arbitrarily complex message exchange patterns (MEP).

Messaging Activities

Activity	Description
CorrelationScope	Creates and configures a CorrelationScope activity that provides implicit management of child messaging activities with a CorrelationHandle object.

Activity	Description
InitializeCorrelation	Creates and configures an InitializeCorrelation activity that is used to initialize correlation without sending or receiving a message.
Receive	Creates and configures a Receive activity that receives a message from a service.
ReceiveAndSendReplyFactory	Creates a pre-configured pair of Send and ReceiveReply activities within a Sequence activity.
Send	Creates and configures a Send activity that sends a message to a service.
SendAndReceiveReplyFactory	Creates a pre-configured pair of Receive and SendReply activities within a Sequence activity.
TransactedReceiveScope	Creates and configures a TransactedReceiveScope activity which enables the flow of transactions into a workflow.

For more information, see http://msdn.microsoft.com/en-us/library/ee829543(v=vs.110).aspx.

Primitives

The .NET Framework provides several system-provided activities that provide a convenient mechanism for performing common tasks.

Activities for Primitives

Activity	Description
Assign	Assigns a value to a variable at the current scope.
Delay	Puts one path of execution into an idle state, possibly allowing the workflow to be unloaded.
InvokeDelegate	Executes a delegate that derives from ActivityDelegate and is exposed as a property.
InvokeMethod	Executes a public method of a CLR object.
WriteLine	Writes a specified string to the console or a specified TextWriter object.

For more information, see http://msdn.microsoft.com/en-us/library/vstudio/ff742828%28v=vs.100%29.aspx.

Runtime

The .NET Framework provides several system-provided activities for accessing the features of the workflow runtime, such as persistence and termination.

Runtime Activities

Activity	Description
NoPersistScope	A container activity that prevents child activities from persisting.

Activity	Description
Persist	Explicitly requests that the workflow persist its data to a durable storage medium (i.e., writing to a file).
TerminateWorkflow	Terminates the running workflow instance.

For more information, see http://msdn.microsoft.com/en-us/library/vstudio/ee358752(v=vs.100).aspx.

Transaction

The .NET Framework has several system-provided activities for modeling transactions, compensation, and cancellation. These programming models allow the workflow to continue forward progress in the event of changes in business logic and error handling.

Transaction Activities

Activity	Description
CancellationScope	Associates cancellation logic, in the form of an activity, with a main path of execution, also expressed as an activity.
CompensableActivity	Supports compensation of its child activities.
Compensate	Explicitly invokes the compensation handler of a CompensableActivity.
Confirm	Explicitly invokes the confirmation handler of a CompensableActivity.
TransactionScope	Demarcates a transaction boundary.
TransactedReceiveScope	Scopes the lifetime of a transaction that is initiated by a received message. The transaction may be flowed into the workflow on the initiating message or created by the dispatcher when the message is received. **Note**: The TransactedReceiveScope is located in the Messaging section of the Toolbox.

For more information, see http://msdn.microsoft.com/en-us/library/vstudio/ee358756(v=vs.100).aspx.

Legacy Workflows

About Legacy Workflows

Beginning with Workflow 2.2, a new object model supports Anthology Student version 17.1 and later. The new object model introduces new namespaces for Anthology Student modules.

Old Namespace	New Namespace

`Cmc.Nexus.Workflow.<modulename>`
Example:
`Cmc.Nexus.Workflow.Sis.Academics`
	`Cmc.Nexus.<modulename>.Workflow`
	Example:
	`Cmc.Nexus.Academics.Workflow`

The new services, namespaces, and entities are documented in the Anthology Student Object Library.

End-of-Life Announcement for Anthology Student Activities (V1)

With the release of Anthology Student 21.0 in October 2019, the EOL date for Anthology Student Activities (V1) is scheduled for October 2020 and the EOS date is scheduled for April 2021. For more information, see End-of-Life for Anthology Student Activities (V1).

New and Migrated Activities

The activities in the toolbox of Workflow Composer are sorted by namespace. Any new activities that have been developed since the introduction of the new object model are added to the corresponding namespaces in the toolbox.

Activities that were developed in the old object model and are required to support events raised out of Anthology Student were migrated to new namespaces.

Example:

The CreateStudentSportsService activity was migrated from `Cmc.Nexus.Workflow.Sis.StudentServices` to `Cmc.Nexus.StudentServices.Workflow`.

If you are creating a new workflow using this activity, use the activity from the new namespace `Cmc.Nexus.StudentServices.Workflow`.

For help about the migrated activity, refer to "CreateStudentSportsService **(V2)**" in the **New Workflows** help section.

Help about the older variant of the activity is found in "CreateStudentSportsService **(V1)**" in the **Legacy Workflows** help section.

The toolbox in Workflow Composer will provide both variants of the CreateStudentSportsService activity until all legacy workflows have been migrated.

The LookupServiceListItem, LookupAreaOfStudy, and LookupListItem activities were not migrated. The functionality of these activities is incorporated into the **LookupReferenceItem** activity in `Cmc.Nexus.Common.Workflow`. Use the LookupReferenceItem activity for any new or migrated workflows.

The LookupGroup activity in `Cmc.Nexus.Workflow` is migrated to LookupStudentGroup in `Cmc.Nexus.Common.Workflow`.

For detailed information about the entities and properties associated with new and migrated activities, refer to the Anthology Student Object Library instead of mapping tables provided in the *Legacy Workflows* help section.

Events

Events raised out of the standard interface for Anthology Student are supported only in the new object model.

Events raised out of the legacy interface for Anthology Student are supported in the legacy model (using legacy contracts, activities, and entity mapping tables). However, the legacy model will be phased out. Any new workflows for events raised out of the legacy interface for Anthology Student 17.1 and later should be migrated to use the new object model.

Contracts

The contracts that the legacy services/activities were developed against are not migrated. Instead, the contracts that the legacy services/activities use become part of the new object model/command model.

The legacy contracts will be supported for a designated length of time allowing for customers to adjust any applicable workflows to use the new entities and their corresponding contracts. The specific steps/process for how affected workflows are updated/modified will need to be determined.

If you are migrating from an older version of Anthology Student to a newer version, you may need to work with two instances of Workflow Composer where one instance uses the V1 and V2 packages of the older Anthology Student version and the second instance uses the V1 and V2 packages for the new Anthology Student version.

When all workflows are migrated to use the new activities, uninstall the old contracts. A new user from Anthology Student 17.1 forward should never install the old contracts/activities.

Converted Entities

In the new object model, the conversion of entity values is no longer required. The CVueIdToPersonIdActivity and PersonIdToCVueIdActivity are no longer needed, and the following conversion formulas no longer apply:

For Student:

- PersonId = (SyStudentId * 10) + 1

Other entities:

- SyStaffId + '2'
- SyAddressId + ‘3’
- PlEmployerContactId + ‘4’

- AmAgencyContactId + ‘5’
- SyOrganizationContactId + ‘6’
- AmOnlineApplicantId + ‘7’

For Student Group: GroupId = (SyGroupsId * 10) + 1

Note: In new and migrated workflows, the Campus (Id) property replaces the Business Unit (Id) property.

End-of-Life for Anthology Student Activities (V1)

On announcing the General Availability (GA) of a major or minor release version of a software product, Anthology Inc. also announces the End-of-Life (EOL) date and End-of-Support (EOS) date for other versions, if applicable. Anthology's policy is to support the newly released GA version as well as the two major or minor release versions immediately preceding the new GA version.

With the release of Anthology Student 21.0 in October 2019, the EOL date for Anthology Student Activities (V1) is scheduled for October 2020 and the EOS date is scheduled for April 2021.

During the EOL period, Anthology will only evaluate Severity 1 issues. All other lesser Severity issues will not be addressed. Once a product version reaches its EOS date, assistance or resolution of any issues reported will no longer be provided. Anthology will only provide the recommendation to upgrade to a version of the product that is not currently EOL or EOS.

The EOL and EOS process allows Anthology to focus development and support efforts on a smaller set of releases, thereby increasing the effectiveness and quality of those releases, while enabling customers to take advantage of the latest available enhancements and resolutions. We encourage our customers, especially those who are on an EOS version or a version in an EOL period, to upgrade to the most current version of our software.

For previous releases of Anthology Student, the Package Manager in Workflow Composer provided Activities for the legacy CampusVue object model (V1) and the new object model (V2).

- Activities that were developed in the legacy object model and are required to support events raised out of Anthology Student were migrated to new namespaces. The migrated activities retain the original activity names and properties but reside in a new namespace.
- Activities that were developed in the legacy object model and are no longer required to support events raised out of Anthology Student were not migrated to new namespaces. Activities that were not migrated are replaced by different activities in the new object model.

Actions Required

Customers using V1 activities in their workflows will need to replace the V1 activities with V2 activities during the EOL period for V1 Activities. The revised workflows will need to be tested to verify the desired functionality.

The table below identifies V1 activities and their corresponding V2 replacements.

- For activities that have been migrated, simply replace the V1 activity with the V2 activity with the same activity name but residing in a different namespace.
- For activities that have not been migrated, remove the V1 activity, insert the suggested V2 activities, and adjust the workflow logic as needed.

V1 Namespaces and Activities	Migrated	V2 Namespaces and Activities
Cmc.Nexus.Converters		
CVueIdToPersonIdActivity	No	GetEntity / SaveEntity in Cmc.Core.Workflow.Activities.EntityModel
PersonIdToCVueIdActivity	No	
Cmc.Nexus.Workflow		
CompleteAction	No	N/A
CreateDocument	Yes	CreateDocument in Cmc.Nexus.Crm.Workflow
LookupExtendedProperty	No	LookupReferenceItem in Cmc.Nexus.Common.Workflow
LookupGroup	No	LookupStudentGroup in Cmc.Nexus.Common.Workflow
LookupListItem	No	LookupReferenceItem in Cmc.Nexus.Common.Workflow
LookupPerson	No	GetEntity in Cmc.Core.Workflow.Activities.EntityModel
LookupPersonDocuments	No	LookupStudentDocuments in Cmc.Nexus.Crm.Workflow
ManageGroupMembership	Yes	ManageGroupMembership in Cmc.Nexus.Common.Workflow
SaveDocument	Yes	SaveDocument in Cmc.Nexus.Crm.Workflow
SaveExtendedProperty	No	SaveEntity in Cmc.Core.Workflow.Activities.EntityModel
SavePerson	No	SaveEntity in Cmc.Core.Workflow.Activities.EntityModel
Cmc.Nexus.Workflow.Crm		**Cmc.Nexus.Crm.Workflow**
CreateTask	Yes	CreateTask
LookupStudentTasks	Yes	LookupStudentTasks
SaveTask	Yes	SaveTask
Cmc.Nexus.Workflow.Sis		**Cmc.Nexus.Common.Workflow**
AssignStudentAdvisor	Yes	AssignStudentAdvisor
LookupAdvisor	Yes	LookupAdvisor
LookupStudent	No	GetEntity in Cmc.Core.Workflow.Activities.EntityModel

V1 Namespaces and Activities	Migrated	V2 Namespaces and Activities
LookupStudentAdvisors	Yes	LookupStudentAdvisors
Cmc.Nexus.Workflow.Sis.Academics		**Cmc.Nexus.Academics.Workflow**
ConvertApplicantToEnrollment	Yes	ConvertApplicantToEnrollment
CreateStudentCourse	Yes	CreateStudentCourse
CreateStudentEnrollmentPeriod	No	N/A
LookupAreaOfStudy	No	LookupReferenceItem in Cmc.Nexus.Common.Workflow
LookupClassSections	Yes	LookupClassSections
LookupCurrentEnrollmentPeriod	Yes	LookupCurrentEnrollmentPeriod
LookupEnrollmentPeriods	Yes	LookupEnrollmentPeriods
LookupTerms	Yes	LookupTerms
SaveStudentCourse	Yes	SaveStudentCourse
SaveStudentEnrollmentPeriod	No	N/A
UpdateNsldsWithdrawalDate	No	N/A
Cmc.Nexus.Workflow.Sis.Academics		**Cmc.Nexus.Common.Workflow**
UpdateStudentStatusToActive	Yes	UpdateStudentStatusToActive
UpdateStudentStatusToDrop	Yes	UpdateStudentStatusToDrop
UpdateStudentStatusToEnrolled	Yes	UpdateStudentStatusToEnrolled
UpdateStudentStatusToGraduate	Yes	UpdateStudentStatusToGraduate
UpdateStudentStatusToLead	Yes	UpdateStudentStatusToLead
UpdateStudentStatusToTempOut	Yes	UpdateStudentStatusToTempOut
Cmc.Nexus.Workflow.Sis.Admissions		Cmc.Nexus.Common.Workflow
UpdateStudentStatusToApplicant	Yes	UpdateStudentStatusToApplicant
Cmc.Nexus.Workflow.Sis.StudentAccounts		**Cmc.Nexus.StudentAccounts.Workflow**
CreateCharge	Yes	CreateCharge
SaveCharge	Yes	SaveCharge
Cmc.Nexus.Workflow.Sis.StudentServices		**Cmc.Nexus.StudentServices.Workflow**
CreateStudentDisabilityDetail	Yes	CreateStudentDisabilityDetail

V1 Namespaces and Activities	Migrated	V2 Namespaces and Activities
CreateStudentSportsService	Yes	CreateStudentSportsService
CreateStudentVeteranDetail	Yes	CreateStudentVeteranDetail
LookupServiceListItem	No	LookupReferenceItem in Cmc.Nexus.Common.Workflow
SaveStudentDisabilityDetail	Yes	SaveStudentDisabilityDetail
SaveStudentSportsService	Yes	SaveStudentSportsService
SaveStudentVeteranDetail	Yes	SaveStudentVeteranDetail

Note: If workflows that contain V1 Activities have not been updated prior to upgrading to Anthology Student 22.x and installing 22.x Activities and Contracts packages, perform the following steps:

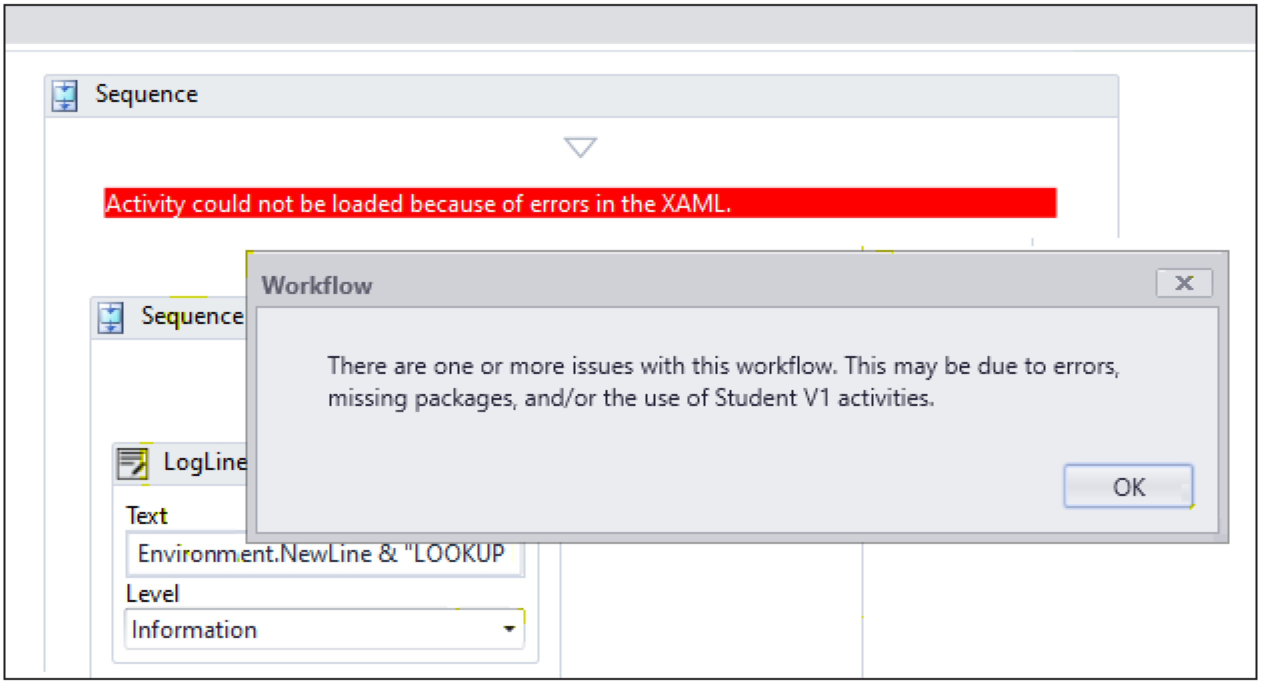
1. Uninstall the V1 and V2 packages for 22.x.
2. Import an earlier version of V1 and V2 packages (e.g., 21.x).
3. Update the workflows to replace the V1 activities.
4. Re-import the 22.x packages.

Run Time Messages About V1 Activities

Workflow Composer 4.x displays warning messages when V1 activities are detected in workflows.

- If no packages with V1 activities are installed —

  When you try to open (from File or Server) a workflow with V1 activities, the following messages are displayed:



Activity could not be loaded because of errors in the XAML.

Workflow

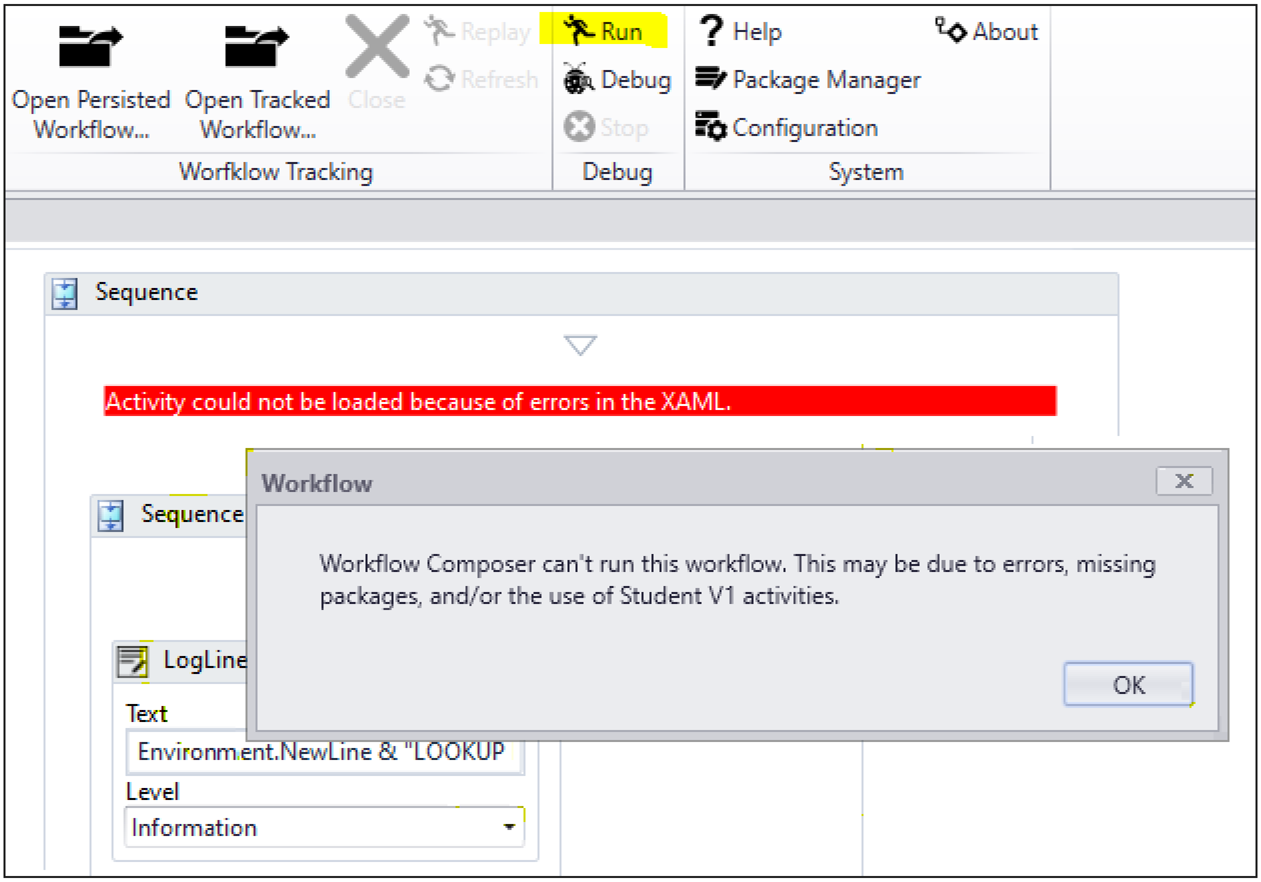
There are one or more issues with this workflow. This may be due to errors, missing packages, and/or the use of Student V1 activities.

OK

Text: Environment.NewLine & "LOOKUP

Level: Information

When you try to run a workflow with V1 activities, the following messages are displayed:



Activity could not be loaded because of errors in the XAML.

Workflow

Workflow Composer can't run this workflow. This may be due to errors, missing packages, and/or the use of Student V1 activities.

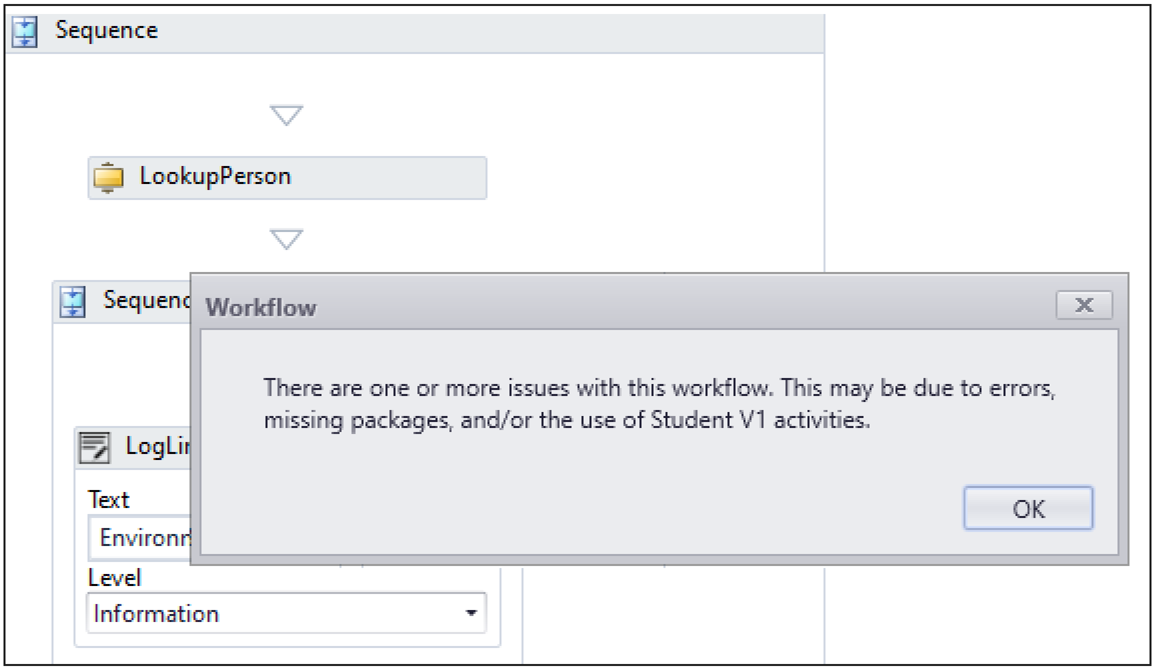
OK

Text: Environment.NewLine & "LOOKUP

Level: Information

- If packages with V1 activities are installed —

When you try to run or open a workflow with V1 activities, the following message is displayed. You can replace the V1 activities and update the workflow.



There are one or more issues with this workflow. This may be due to errors, missing packages, and/or the use of Student V1 activities.

Note: If you imported packages for Anthology Student 22.0 or later, you will need to install a prior version of Anthology Student packages to edit workflows that have V1 activities. After you have edited the workflows and replaced V1 with V2 activities, you can re-import the newer packages.

Script to Locate V1 Activities

To locate all occurrences of V1 activities in your workflows, you can run the script below. The script identifies the workflows using V1 activities and, on a per-workflow basis, lists the V1 activities that are being used. It also gives a count of how many times each activity occurs in the workflow so that you know how many occurrences to look for in the workflow while updating it.

If you wish to validate the script in an environment, you can do so by opening the XAML for the workflow in a text editor and searching for all occurrences of V1 activity elements in the document. These elements will have the following namespace prefixes:

- `cnc`
- `cnw`
- `cnwc`
- `cnws`
- `cnwsa`
- `cnwsa1`
- `cnwss`
- `cnwss1`

Example:

```
<cnw:LookupGroup DisplayName="Lookup Pending App Group" Group="[groupadd]"
GroupId="314411" sap2010:WorkflowViewState.IdRef="LookupGroup_1" />
```

LookupGroup is a V1 activity in the XAML because it starts with "`cnw:`" which is the Cmc.Nexus.Workflow namespace.

The results of the manual search should match the results shown by the script in terms of what activities are identified, and how many of each there are.

```
/*******************************************************************
**
** Find Workflows Using V1 Activities
**
** Author: Mike Carter, Technical Account Manager, Anthology Inc
** Date: 7/16/2021
**
** Locate enabled workflows which are using V1 activities and,
** on a per-workflow basis, list each Activity name and how
** many occurrences of each activity there are in the workflow
**
** Identify the V1 activites by the following namespaces:
**    Cmc.Nexus.Converters
**    Cmc.Nexus.Workflow (and also namespaces prefixed by this)
**
********************************************************************/

declare
        @WorkflowName nvarchar(max)
       ,@EnabledVersion int
       ,@xaml xml

if object_id('tempdb..#AffectedWorkflows') is not null drop table #AffectedWorkflows
select
        WorkflowDefinition.Name as WorkflowName
       ,WorkflowDefinitionVersion.Revision as EnabledVersion
       ,cast(WorkflowDefinitionVersion.Xaml as XML) as xaml

into #AffectedWorkflows

from
       WorkflowDefinition
       inner join WorkflowDefinitionVersion on WorkflowDefinitionVersion.WorkflowDefinitionId = WorkflowDefinition.Id

where
       WorkflowDefinitionVersion.IsEnabled = 1
       and (
               WorkflowDefinitionVersion.Xaml like '%clr-namespace:Cmc.Nexus.Converters%'
               or WorkflowDefinitionVersion.Xaml like '%clr-namespace:Cmc.Nexus.Workflow%'
       )

--select * from #AffectedWorkflows order by WorkflowName

if object_id('tempdb..#Output') is not null drop table #Output
create table #Output (WorkflowName nvarchar(max), EnabledVersion int, ActivityName varchar(50),
Occurrences int)
```

```
declare Records_Cursor cursor local fast_forward for
       select WorkflowName, EnabledVersion, xaml from #AffectedWorkflows
open Records_Cursor

while 1=1
begin
       fetch next from Records_Cursor INTO @WorkflowName, @EnabledVersion, @xaml
       if @@FETCH_STATUS <> 0
               break

       ;with walkXML
       as
       (       select
                        startNodes.query('./*') curLevelXml
                       ,startNodes.value('local-name(.)', 'varchar(50)') NodeName
                       ,startNodes.value('namespace-uri(.)', 'varchar(500)') NodeNamespaceUri
               from
                       @xaml.nodes('/*') t(startNodes) --starting with nodes under the root

               union all

               select
                        childNodes.query('./*') curLevelXml
                       ,childNodes.value('local-name(.)', 'varchar(50)') NodeName
                       ,childNodes.value('namespace-uri(.)', 'varchar(500)') NodeNamespaceUri
               from
                       walkXML
                       cross apply curLevelXml.nodes('./*') t2(childNodes) --child nodes descending down
xml document
       )

       insert into #Output(WorkflowName, EnabledVersion, ActivityName, Occurrences)
       select @WorkflowName, @EnabledVersion, NodeName, count(1)
       from walkXML
       where
               NodeNamespaceUri like 'clr-namespace:Cmc.Nexus.Converters%'
               or NodeNamespaceUri like 'clr-namespace:Cmc.Nexus.Workflow%'
       group by NodeName

end
close Records_Cursor;
deallocate Records_Cursor;


select * from #Output
order by WorkflowName, ActivityName
```

Entity Mapping

Anthology implements a new domain model that aggregates the entities from the three legacy application domains into a single unified model. For example, the Anthology domain includes a Person entity. The Student and Staff entities in Anthology Student will map to the Person entity. The Contact entity in CRM will map to the Person entity. The Donor entity in Talisma Fundraising will map to the Person entity. Additionally, the Anthology domain includes functional roles. The end result is that there is a common Person entity which has associated functional roles.

Common Entity Properties

The common entity properties `OriginalValues` and `ModifiedProperties` are only initialized for use in events when `EntityState` is Modified.

`ExtendedProperties` is not currently used by any events.

Converted Entities

Entities that are mapped between Anthology Student and the Anthology domain are marked with the keyword CONVERTED in the mapping tables. The following conversion formula applies to the converted entities:

For Student:

- PersonId = (SyStudentId * 10) + 1

Other entities:

- SyStaffId + '2'
- SyAddressId + ‘3’
- PlEmployerContactId + ‘4’
- AmAgencyContactId + ‘5’
- SyOrganizationContactId + ‘6’
- AmOnlineApplicantId + ‘7’

Class-based Inheritance

Some classes in the Cmc.Nexus.Sis.FinancialAid entity inherit properties of another class. When one class inherits from another, all fields from the base class are also available.

Mapping Tables

Refer to the following topics for the mapping of Anthology entities and their associated classes and properties to tables and fields in the Anthology Student database.

Cmc.Nexus

The following table shows the mapping of classes and properties in the Cmc.Nexus entity to tables and fields in the Anthology Student database.

Cmc.Nexus Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
AddressBase			
	AddressTypeId	N/A	This property is required in the contract; however, the current mapping logic ignores the value provided since the only address attributes currently being updated are those on the SyStudent record.
	City	SyStudent.City, SyStaff.City, SyAddress.City	Depends on if SyStudent, SyStaff, or SyAddress record is in context.
	CountryId	SyStudent.SyCountryId, SyStaff.SyCountryId, SyAddress.SyCountryId	Depends on if SyStudent, SyStaff, or SyAddress record is in context.
	CountryName	SyAddress.Country	N/A for SyStudent and SyStaff records
	CountyId	SyStudent.SyCountyId, SyAddress.SyCountyId	Depends on if SyStudent or SyAddress record is in context. N/A for SyStaff
	CountyName	SyAddress.County	N/A for SyStudent and SyStaff records
	DoNotContact	N/A	
	DoNotContactOverride	N/A	
	EffectiveBeginDate	SyAddress.BeginDate	N/A for SyStudent and SyStaff records
	EffectiveEndDate	SyAddress.EndDate	N/A for SyStudent and SyStaff records
	FirstName	N/A	
	Id	SyAddress.SyAddressId or NULL	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	IsNotValid	SyStudent.BadAddr	N/A for SyStaff and SyAddress records
	IsPreferred	N/A	
	IsSeasonal	SyAddress.Yearly	N/A for SyStudent and SyStaff records
	LastName	SyAddress.LastName	N/A for SyStudent and SyStaff records
	Note	N/A	
	PhoneNumber	SyStudent.Phone, Systaff.Phone, SyAddress.Phone	Depends on if SyStudent, SyStaff, or SyAddress record is in context.
	PostalCode	SyStudent.Zip, SyStaff.Zip, SyAddress.Zip	Depends on if SyStudent, SyStaff, or SyAddress record is in context.
	StateId	N/A	
	StateName	SyStudent.State, SyStaff.State, SyAddress.State	Depends on if SyStudent, SyStaff, or SyAddress record is in context.
	StreetAddress	SyStudent.Addr1, SyStaff.Addr1, SyAddress.Addr1	Depends on if SyStudent, SyStaff, or SyAddress record is in context.
	TitleId	SyAddress.TitleID	N/A for SyStudent and SyStaff records
BusinessUnit			
	Id	SyCampus.SyCampusId	
Ethnicity			
	Id	SyStudentAmRace.AmRaceId	
FunctionalRole			
	RoleType	N/A	An enum property. 0= Unknown, 1= Prospect, 2=Student, 3=Staff

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
Group	The Group class in Anthology maps to the SyGroups table in Anthology Student. Eventually, the SyStaffGroup and SyEmpGroups tables will also be mapped to this entity. The Anthology domain merges Staff Groups, Student Groups, and Employer Groups into one Group and Group Membership class. This brings consistency to all functionality for the Groups concept in Anthology. Additionally, Groups in Anthology are able to mix membership between Person and Organizations within a single Group.		
	AdvisorRelationshipTypeId	SyStaffGroup.AdvisorModule	
	AssociatedBusinessUnits	SyGroups.SyCampusGrpId, SyCampusList.SyCampusId	
	Code	SyGroups.Code	
	ExpirationDate	SyGroups.DateExpires	
	Id	SyGroups.SyGroupsId	Mapping occurs between Anthology Student and Anthology: For Student Group: GroupId = (SyGroupsId * 10) + 1
	IsActive	SyGroups.Active	
	IsPublic	SyGroups.PublicGroup	
	IsStaffGroup	N/A or True if mapping from Staffgroup	No mapping as all Staff Groups in Anthology Student are stored in SyStaffGroup; however, the value in this Anthology property determines which table gets updated in Anthology Student. If this is True, then SyStaffGroup is updated.
	IsSystem	N/A	
	MembershipFunctionalRoles	N/A	No mapping. Anthology Group entity allows membership in a group from different entities.
	Name	SyGroups.Descrip	
	OwnerUserId	SyGroups.SyStaffId	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	StaffGroupBusinessUnitId	N/A	No mapping. Method for determining which Business units (Campuses in Anthology Student) a given staff user is associated with is implemented differently in Anthology than in Anthology Student.
	StaffGroupType	N/A	
	Usage	N/A	
GroupMembership			
	AddedDate	SyStudGrp.DateAdded	
	AddedUserId	StudGrp.UserIdOn	
	GroupId	SyGroups.SyGroupsId	Mapping occurs between Anthology Student and Anthology: For Student Group: GroupId = (SyGroupsId * 10) + 1
	Id	SyStudGrp.SyStudGrpId	
	IsActive	SyStudGrp.Active	
	OrganizationId	N/A	
	PersonId	SyStudGrp.SyStudentId (CONVERTED)	
	RemovedDate	SyStudGrp.DateOff	
	RemovedUserId	SyStudGrp.UserIdOff	
Nationality			
	Id	SyStudent.AmNationalityId	
Organization			No mapping is currently done for this class.

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	Addresses	N/A	
	DateCreated	N/A	
	Id	N/A	
	Name	N/A	
	Note	N/A	
	OrganizationContacts	N/A	
	OrganizationUrl	N/A	
	OwnerId	N/A	
	Phones	N/A	
	PrimaryContactId	N/A	
	SicCode	N/A	
Person			
	Addresses	Collection of PersonAddress	
	BirthCountryId	N/A	
	BirthDate	SyStudent.Dob	
	Cases	N/A	
	Emails	Collection of PersonEmail	
	Ethnicities	Collection of PersonEthnicity	
	FirstName	SyStudent.FirstName, SyStaff.FirstName	Depends on if SyStudent or SyStaff record is in context.
	FunctionalRoles	See FunctionalRole	Students, Prospects are added when mapping from SyStudent.
	GenderId	SyStudent.AmSexId	
	HasDisability	SyStudent.Disabled	
	Id	SyStudent.SyStudentId (CONVERTED), SyStaff.SyStaffId (CONVERTED)	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	Interactions	Collection of Interaction (see Interaction class in CMC.Nexus.Crm)	
	Interests	N/A	
	LastContactDate	SyStudent.LastActivityDate	
	LastName	SyStudent.LastName, SyStaff.LastName	Depends on if SyStudent or SyStaff record is in context.
	MaidenName	SyStudent.MaidenName	
	MaritalStatusId	SyStudent.AmMaritalId	
	MiddleName	SyStudent.MiddleName	
	Name	N/A	
	Nationalities	Collection of Nationality where Id field maps to SyStudent.AmNationalityId.	Contract supports multiple values; however, only the first value provided is updated to SyStudent.AmNationalityId.
	NickName	SyStudent.NickName	
	Phones	Collection of PhoneBase	
	PreferredLanguageId	N/A	
	Prospects	Read-only collection of Prospect	This is a read-only collection. No data provided in this collection will be persisted to the Anthology Student database. See Prospect class in CMC.Nexus.SIS.Admissions for additional information.
	Salutations	N/A	
	Ssn	SyStudent.Ssn	
	Students	Read-only collection of Student	This is a read-only collection. No data provided in this collection will be persisted to the Anthology Student database. See Student class in CMC.Nexus.SIS for additional information.
	SuffixId	SyStudent.AmSuffixId	Not mapped in Saved events.

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	TitleId	SyStudent.AmTitleId	
	Veteran	SyStudent.Vet	
PersonAddress			Inherits from AddressBase.
PersonDocument			
	ApprovalDate	CmDocument.DateApproved	
	AwardYear	CmDocument.AwardYear	
	CreatedbyUserId	CmDocument.AddUserId	Documents added via automated Document Scheduler processes may not contain a value for this field.
	DocumentCategoryId	CmDocument.SyModuleId	
	DocumentStatusId	CmDocument.CmDocStatusId	
	DocumentTypeId	CmDocument.CmDocTypeId	
	DueDate	CmDocument.DateDue	
	ExpirationDate	CmDocument.DateExpires	
	Id	CmDocument.CmDocumentId	
	ModifiedByUserID	CmDocument.UserId	Documents added via automated Document Scheduler processes may not contain a value for this field.
	Note	CmDocument.Comments	
	PersonId	CmDocument.SyStudentId (CONVERTED)	
	ProspectId	CmDocument.SyStudentId	
	ReceivedDate	CmDocument.DateRecv	
	RequestDate	CmDocument.DateReq	
	SentDate	CmDocument.DateSent	
	StudentId	CmDocument.SyStudentId	StudentId and ProspectId are purposely both mapped to CmDocument.SyStudentId
	WorkflowInstanceId	CmDocument.WorkflowInstanceId	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
PersonEmail			
	DisplayName	N/A	
	EmailAddress	SyStudent.Email, SyStaff.Email, SyStaff.Email_ReplyTo	Depends on if SyStudent or SyStaff record is in context.
	EmailTypeId	If SyStaff.Email, set to "1". If SyStaff.Email_ReplyTo, set to "2".	EmailType is enum for now: 1 = PRIMARY, 2 = SECONDARY
PersonEthnicity			
	Ethnicities	Collection of Ethnicity where Id field maps to SyStudentAmRace.AmRaceId.	Multiple values for Ethnicity can be provided.
	IsHispanicLatino	SyStudent.IsHispanic	
	PersonId	SyStudentAmRace.SyStudentId (CONVERTED)	
PersonPhone			Inherits from PhoneBase
PhoneBase			
	DoNotContact	N/A	
	DoNotContactOverride	N/A	
	Extension	N/A	There is an Ext for Work phone, but not Phone in SyStudent.
	IsNotValid	SyStudent.Badphone	
	IsPreferred	N/A	
	PhoneNumber	SyStudent.Phone, SyStaff.Phone, SyStaff.WorkFaxPhone, SyStaff.WorkPhone, SyStaff.HomePhone	Depends on if SyStudent or SyStaff record is in context.
	PhoneTypeId	If SyStaff.Phone, set to "1". If SyStaff.WorkFaxPhone, set to "2". If SyStaff.CellPhone, set to "3". If SyStaff.HomePhone, set to "4"	PhoneType is enum for now: 1=HOME, 2=WORK, 3=MOBILE, 4=OTHER

Cmc.Nexus.Crm

The following table shows the mapping of classes and properties in the Cmc.Nexus.Crm entity to tables and fields in the Anthology Student database.

Cmc.Nexus.Crm Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
Interaction	No mapping currently exists to any Anthology Student table.		
	BusinessUnitId		
	CaseId		
	CaseState		
	CommunicationChannelId		
	CommunicationDirection		
	CreatedbyUserId		
	CreatedDate		
	EventHeader		
	EventType		
	From		
	Id		
	InteractionEmail		
	MessageBody		
	PersonId		
	PhoneNumber		
	ProspectId		
	ShowExpandCollapse		
	Subject		
	To		
Task			
	CreatedByUserId	CmEvent.SetupBy	
	DueDate	CmEvent.DueDate	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	Id	CmEvent.CmEventId	
	Location	N/A	
	Note	CmEvent.Comments	
	OwnerUserId	CmEvent.SyStaffId (CONVERTED)	
	People	People is a collection of Person. CmEvent.SyStudentId (CONVERTED)	PersonId is the only property that is populated.
	PercentageComplete	N/A	
	Priority	CmEvent.Priority	Converted to enum TaskPriority
	ReminderDate	CmEvent.RemindDate	
	ReminderInterval	N/A	
	StartDate	CmEvent.StartDate	The time the activity is scheduled to begin. Only the time portion of this value is relevant.
	Subject	CmEvent.Subject or EmailSubject if Subject is NULL	
	TaskResultId	CmEvent.CmEventResultId	
	TaskStatusId	CmEvent.CmEventStatusId	
	TaskTypeId	CmEvent.CmTemplateId	
	WorkflowInstanceId	CmEvent.WorkflowInstanceId	

Cmc.Nexus.FinancialAid.Services

The following table shows the mapping of classes and properties in the Cmc.Nexus.FinancialAid.Services namespace to tables and fields in the Anthology Student database.

Cmc.Nexus.FinancialAid.Services Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
GetIsirResponse	This contract specifies the ServiceResponse from the service operation called when the LookupIsir activity is executed. This contract is coupled to the response returned from the GetIsirResponse service operation. **Note**: GetIsirResponse is a custom service response message and not an entity within the Anthology command model. The GetIsirResponse fields are returned in the `OutArgument<IsirMessage>` of the LookupIsir activity.		
	AdditionalFields		This is of type Dictionary and will be a key/value pair array that holds the data for all fields from the ISIR all data view (vw_FaIsirNewAllIncluded) that are not separate defined properties in the contract.
	ApplicationCompletedDate	vw_FaIsirNewAllIncluded.DateCompleted	
	ApplicationReceiptDate	vw_FaIsirNewAllIncluded.ApplicationRcptDate	
	AwardYearIdentifier	vw_FaIsirNewAllIncluded.FaYearId	
	CommentCodes	vw_FaIsirNewAllIncluded.CommentCodes	Mapping logic would parse vw_FaisirNewAllIncluded.CommentCodes and build collection of CommentCodes.
	DegreeCertificate	vw_FaIsirNewAllIncluded.Degree	
	DependencyStatus	vw_FaisirNewAllIncluded.Model	
	EnrollmentStatus	FaStudentPell.PellEnrollmentStatus	
	FatherIncome	vw_FaIsirNewAllIncluded.FatherIncome	
	GradeLevel	vw_FaIsirNewAllIncluded.CollegeGradeLevel	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	HasChildrenToSupport	vw_FaIsirNewAllIncluded.Children	
	HasDrugConviction	vw_FaIsirNewAllIncluded.DrugOffense	
	HasHighSchoolDiplomaGed	vw_FaIsirNewAllIncluded.HSGedReceived	
	HasOtherLegalDependents	vw_FaIsirNewAllIncluded.LegalDependents	
	InstitutionalEfc	FaStudentPell.InstitutionalEfc	
	InterestedWorkStudyStudentLoans	vw_FaIsirNewAllIncluded.InterestedInAid	
	IsActiveDutyArmedForces	vw_FaIsirNewAllIncluded.ActiveDutyMilitary	
	IsAutomaticZeroEfc	vw_FaIsirNewAllIncluded.Auto0EFCFlag	
	IsDodMatch	vw_FaIsirNewAllIncluded.DodMatchFlag	
	IsFirstBachelorDegree	vw_FaIsirNewAllIncluded.FirstBachDegree	
	IsirMatchId	FaIsirStudentMatch.FaIsirStudentMatchId	
	IsirReceivedDate	vw_FaIsirNewAllIncluded.DateAdded	
	IsirSummaryId	vw_FaIsirNewAllIncluded.FaisirMainId	
	IsSimplifiedNeedsTestMet	vw_FaIsirNewAllIncluded.SimplifiedNeeds	
	IsStudentMale	vw_FaisirNewAllIncluded.Male	
	IsStudentMarried	vw_FaIsirNewAllIncluded.StudentMaritalStatusAsOfToday	
	IsVeteranArmedForces	vw_FaIsirNewAllIncluded.Veteran	
	IsWorkingTowardsMastersDoctorate	vw_FaIsirNewAllIncluded.DegreeBeyond	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	MotherIncome	vw_FaIsirNewAllIncluded.MotherIncome	
	NsldsActiveBankruptcyFlag	vw_FaIsirNewAllIncluded.ActiveBankruptcyFlag	
	NsldsAggregateLoanBalance	vw_FaIsirNewAllIncluded.AggrCombinedBal	
	NsldsAggregateSubLoanBalance	vw_FaIsirNewAllIncluded.AggrSubsidizedBal	
	NsldsAggregateUnsubLoanBalance	vw_FaIsirNewAllIncluded.AggrUnsubBal	
	NsldsDatabaseResultsFlag	vw_FaIsirNewAllIncluded.DatabaseResultsFlag	
	NsldsDefaultedLoanFlag	vw_FaIsirNewAllIncluded.DefaultedLoanFlag	
	NsldsDischargedLoanFlag	vw_FaIsirNewAllIncluded.DischargedLoanFlag	
	NsldsFraudLoanFlag	vw_FaIsirNewAllIncluded.FraudLoanFlag	
	NsldsPellLifetimeEligibilityUsed	vw_FaIsirNewAllIncluded.PellLifetimeEligUsed	
	NsldsPellLifetimeLimitFlag	vw_FaIsirNewAllIncluded.PellLifeTimeLimitFlag	
	NsldsPellOverpaymentFlag	vw_FaIsirNewAllIncluded.PellOverpayFlag	
	NsldsPerkinsOverpaymentFlag	vw_FaIsirNewAllIncluded.PerkinsOverpayFlag	
	NsldsSatisfactoryRepaymentFlag	vw_FaIsirNewAllIncluded.LoanSatisfactoryRepaymentFlag	
	NsldsSeogOverpaymentFlag	vw_FaIsirNewAllIncluded.SeogOverpayFlag	
	NsldsTeachOverpaymentFlag	vw_FaIsirNewAllIncluded.TeachOverpayFlag	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	NsldsUnusualEnrollmentFlag	vw_FaIsirNewAllIncluded.EnrollmentPatternFlag	
	ParentAdjustedGrossIncome	vw_FaIsirNewAllIncluded.ParentGross	
	ParentBusinessFarmNetWorth	vw_FaIsirNewAllIncluded.ParentBusiness	
	ParentCash	vw_FaIsirNewAllIncluded.ParentCash	
	ParentChildSupportPaid	vw_FaIsirNewAllIncluded.ParentChildSupportPaid	
	ParentChildSupportReceived	vw_FaIsirNewAllIncluded.ParentChildSupportReceive	
	ParentCombatPay	vw_FaIsirNewAllIncluded.ParentCombatPay	
	ParentContribution	vw_FaIsirNewAllIncluded.ParentContribution	
	ParentDislocatedWorker	vw_FaIsirNewAllIncluded.ParentDislocatedWorker	
	ParentEducationCredits	vw_FaIsirNewAllIncluded.ParentEducationCredits	
	ParentEligibletoFile1040	vw_FaisirNewAllIncluded.ParentElig1040	
	ParentFederalBenefitsFreeSchoolLunch	vw_FaIsirNewAllIncluded.ParentFreeLunch	
	ParentFederalBenefitsSsi	vw_FaIsirNewAllIncluded.ParentSSIBenefits	
	ParentFederalBenefitsSnap	vw_FaIsirNewAllIncluded.ParentFoodStamps	
	ParentFederalBenefitsTanf	vw_FaIsirNewAllIncluded.ParentTANFBenefits	
	ParentFederalBenefitsWic	vw_FaIsirNewAllIncluded.ParentWICBenefits	
	ParentIncomeTaxPaid	vw_FaIsirNewAllIncluded.ParentIncomeTax	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	ParentInterestIncome	vw_FaIsirNewAllIncluded.ParentInterestIncome	
	ParentInvestmentNetWorth	vw_FaIsirNewAllIncluded.ParentInvestment	
	ParentIraDistributions	vw_FaisirNewAllIncluded.ParentIRADistributions	
	ParentIraPayments	vw_FaIsirNewAllIncluded.ParentIRAPayments	
	ParentLegalResidenceDate	vw_FaIsirNewAllIncluded.ParentLegResDate	
	ParentLegalStateOfResidence	vw_FaIsirNewAllIncluded.ParentLegState	
	ParentMaritalStatus	vw_FaIsirNewAllIncluded.ParentMaritalStatus	
	ParentMilitaryClergyAllowance	vw_FaIsirNewAllIncluded.ParentMilitaryAllowance	
	ParentNeedBasedEmployment	vw_FaIsirNewAllIncluded.ParentNeedBasedEmployment	
	ParentNumberInCollege	vw_FaIsirNewAllIncluded.ParentNumCollege	
	ParentNumberInFamily	vw_FaIsirNewAllIncluded.ParentNumFamily	
	ParentNumberOfExemptions	vw_FaIsirNewAllIncluded.ParentExemptions	
	ParentPensionBenefits	vw_FaIsirNewAllIncluded.ParentPensionPayments	
	ParentTaxFormUsed	vw_FaisirNewAllIncluded.ParentTaxFormType	
	ParentTaxReturnStatus	vw_FaIsirNewAllIncluded.ParentTaxReturnFilingStatus	
	ParentUntaxedIncomeOther	vw_FaIsirNewAllIncluded.ParentOtherUntaxedIncome	
	ParentUntaxedIncomeTotal	vw_FaIsirNewAllIncluded.ParentUntaxedIncomeTotal	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	ParentUntaxedPension	vw_faisirNewAllIncluded.ParentUntaxedPension	
	ParentVeteranNonEducationBenefits	vw_faisirnewAllIncluded.ParentvetNonEducationBenefits	
	PellGrantAmount	FaStudentPell.PellAmount	
	PellGrantEligibilityFlag	vw_FaIsirNewAllIncluded.PellEligFlag	
	PellPaidEfc	FaStudentPell.PellPaidEfc	
	PrimaryEfc	vw_FaIsirNewAllIncluded.PEFC	
	SarCCode	vw_FaIsirNewAllIncluded.SarCFlag	
	SelectedForVerification	vw_FaIsirNewAllIncluded.SelectedForVerification	
	SpouseIncome	vw_FaIsirNewAllIncluded.SpouseIncome	
	StudentAdjustedGrossIncome	vw_FaIsirNewAllIncluded.StudentGross	
	StudentCitizenship	vw_FaIsirNewAllIncluded.Citizen	
	StudentDateOfBirth	vw_FaIsirNewAllIncluded.DOB	
	StudentId	FaIsirStudentMatch.SyStudentId	
	StudentIncome	vw_FaisirNewAllIncluded.StudentIncome	
	StudentLegalResidenceDate	vw_FaIsirNewAllIncluded.StudentLegResDate	
	StudentLegalStateOfResidence	vw_FaIsirNewAllIncluded.StudentLegState	
	StudentMaritalStatus	vw_FaIsirNewAllIncluded.StudentMaritalStatus	
	StudentMaritalStatusDate	vw_FaIsirnewAllIncluded.StudentMaritalStatusDate	
	StudentTaxFormUsed	vw_FaIsirNewAllIncluded.StudentTaxFormType	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	StudentTaxReturnStatus	vw_FaIsirNewAllIncluded.StudentTaxReturnFilingStatus	
	TransactionProcessDate	vw_FaIsirNewAllIncluded.TransactionProcessedDate	
	TransactionReceiptDate	vw_FaIsirNewAllIncluded.TransactionReceiptDate	
	VerificationStatus	FaStudentPell.VerifStatus	
	VerificationTrackingFlag	vw_FaIsirNewAllIncluded.VerificationTrackingFlag	
IsirMatch			
	AwardYearId	FaIsirStudentMatch.FaYearId	
	CreatedDateTime	FaIsirStudentMatch.DateAdded	
	Id	FaIsirStudentMatch.FaIsirStudentMatchId	
	IsirSummaryId	FaIsirStudentMatch.FaIsirMainId	
	LastModifiedDateTime	FaIsirStudentMatch.DateLstMod	
	LastModifiedUserId	FaisirStudentMatch.UserId	
	RowVersion	N/A	
	SchoolCode	FaIsirStudentMatch.PellId	
	StudentId	FaIsirStudentMatch.SyStudentId	

Cmc.Nexus.Sis

The following table shows the mapping of classes and properties in the Cmc.Nexus.Sis entity to tables and fields in the Anthology Student database.

Cmc.Nexus.Sis Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
Staff			
	AdmissionsRepTypeId	SyStaff.AmReptypeId	
	CampusGroupId	SyStaff.SyCampusGrpId	
	Code	SyStaff.Code	
	Department	SyStaff.Department	
	HiredDate	SyStaff.HiredDate	
	Id	SyStaff.SyStaffId	
	IsActive	SyStaff.Active	
	Note	SyStaff.Comments	
	PersonId	SyStaff.SyStaffId (CONVERTED)	
	Position	SyStaff.Position	
	TaskPolicyyId	SyStaff.CmPolicyId	
	Title	SyStaff.Title	
StaffGroup			
	AdvisorModule	SyStaffGroup.AdvisorModule	
	Code	SyStaffGroup.Code	
	Id	SyStaffGroup.SyStaffGroupId	
	IsActive	SyStaffGroup.Active	
	IsSystemCode	SyStaffGroup.System	
	Name	SyStaffGroup.Descrip	
StaffGroupMember			
	Id	SyStaffByGroup.SyStaffByGroupId	
	StaffGroupId	SyStaffByGroup.SyStaffGroupId	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	StaffId	SyStaffByGroup.SyStaffId	
Student			
	AssociatedBusinessUnits	AdEnroll.SyCampusId	The contract allows for multiple Business Unit IDs. Anthology Student has only a single Campus ID that is populated in this property.
	AthleticIdentifier	SyStudent.AthleticId	
	Id	AdEnroll.SyStudentId	
	PersonId	AdEnroll.SyStudentId (CONVERTED)	
	ShiftId	AdEnroll.AdShiftId	
	StudentEnrollmentPeriods	Collection of StudentEnrollmentPeriod	When the Student Enrollment Wizard uses a Person Saving event, each step only fills out a few fields in the Person.Students(0).StudentEnrollmentPeriods(0) entity based on the step Context.
	StudentExtraCurriculars	Collection of StudentExtraCurricular	
	StudentNumber	AdEnroll.Stunum	
StudentAdvisor	The StudentAdvisor entity is created for the sole purpose of supporting the current domain of Anthology Student. In the long term vision, student advisors will be persisted as Relationships. The specific members of the Relationship class in Anthology as well as all of the details around the Relationships feature in general have not yet been finalized. Thus it is premature to use the Relationship entity and contract to support the needed workflow functionality for Advisors in Anthology Student 17.0. For now, the StudentAdvisor class/entity is available and aligned completely with the existing Anthology Student domain.		
	AdvisorModule	SyAdvisorByEnroll.AdvisorModule	
	Id	SyAdvisorByEnroll.SyAdvisorByEnrollId	
	StaffGroupId	SyAdvisorByEnroll.SyStaffGroupId	
	StaffId	SyAdvisorByEnroll.SyStaffId	
	StudentEnrollmentPeriodId	SyAdvisorByEnroll.AdEnrollId	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
StudentExtraCurricular			
	ExtraCurricularId	AmprospectExtraCurr.AmExtraCurrId	
	IsPrimary	AmprospectExtraCurr.PrimaryExtraCurr	
	StudentId	AmprospectExtraCurr.SyStudentId	
Anthology Class	**Anthology Property**	**Anthology Student Table.Field Name**	**Comments**
Students	AssociatedBusinessUnits	AdEnroll.SyCampusId	The contract allows for multiple business unit IDs. Anthology Student has only a single Campus ID that is populated in this property.
	Id	AdEnroll.AdEnrollId	
	PersonId	AdEnroll.SyStudentId (CONVERTED)	Mapping occurs between Anthology Student and Anthology. For Student, PersonId = (SyStudentId * 10) + 1. Other entities: SyStaffId + '2', SyAddressId + ‘3’,PlEmployerContactId + ‘4’,AmAgencyContactId + ‘5’,SyOrganizationContactId + ‘6’,AmOnlineApplicantId + ‘7’
	ShiftId	AdEnroll.AdShiftId	
	StudentEnrollmentPeriods	See StudentEnrollmentPeriod class in CMC.Nexus.Sis.Academics.	
	StudentNumber	AdEnroll.Stunum	

Cmc.Nexus.Sis.Academics

The following table shows the mapping of classes and properties in the Cmc.Nexus.Sis.Academics entity to tables and fields in the Anthology Student database.

Cmc.Nexus.Sis.Academics Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
AreasOfStudy	Current mapping logic only updates to the AdProgramVersion table. No mapping is currently done to the AdConcentration table.		
	AreaOfStudyType	N/A	
	Code	AdprogramVersion.Code	
	GradeScaleId	AdProgramVersion.AdGradeScaleId	
	Id	AdProgramVersion.AdProgramversionId	
	IsActive	AdprogramVersion.Active	
	MinimumGpa	N/A	
	Name	AdProgramVersion.Descrip	
	ProgramId	AdProgramVersion.AdProgramId	
	RequiredCredits	AdProgramVersion.CreditsReq	
	RequiredHours	AdProgramVersion.HoursReq	
ClassSection			
	AddDropDate	AdClassSched.AddDropDate	
	AllowWaitlist	AdClassSched.AllowWaitlisting	
	AuditAdvisementRequired	AdClassSched.AuditAdvisementRequired	
	AutoDropConsecutiveHoursAbsent	AdClassSched.DropConsAbsent	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	AutoDropCumulativeHoursAbsent	AdClassSched.DropCumAbsent	
	AutoDropEnforceAfterLastDateToWithdraw	AdClassSched.EnforceAttendanceLDW	
	AutoDropPercentageHoursAbsent	AdClassSched.DropAbsentPct	
	AutoWarningConsecutiveHoursAbsent	AdClassSched.WarnConsAbsent	
	AutoWarningCumulativeHoursAbsent	AdClassSched.WarnCumAbsent	
	AutoWarningOnClassRosterAfterLastDateToWithdraw	AdClassSched.AutoDropWarningForLDW	
	AutoWarningPercentageHoursAbsent	AdClassSched.WarnAbsentPct	
	BusinessUnits	Collection of BusinessUnit	AdClassSched.SyCampusId will be the only value populated in this collection.
	ClassSectionInstructors	Collection of ClassSectionInstructor	
	Course	See Course class.	
	CreatedbyUserId	AdClassSched.UserId	
	DeliveryMethodId	AdClassSched.AdDeliveryMethodId	
	EndDate	AdClassSched.EndDate	
	Id	AdClassSched.AdClassSchedId	
	IsActive	AdClassSched.Active	
	LastDayToWithdrawDate	AdClassSched.LdwDate	
	LmsExtractStatus	AdClassSched.LmsExtractStatus	
	MakeupMaxType	AdClassSched.MakeupMaxtype	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	MakeupMaxValue	AdClassSched.MakeupMaxNum	
	MaximumSeats	AdClassSched.MaxStudents	
	ModifiedByUserId	AdClassSched.UserId	
	Note	AdClassSched.SchedComment	
	PassFailType	AdClassSched.PassFailSetting	
	PostAttendancetype	AdClassSched.Attendancetype	
	RegisteredStudents	AdClassSched.RegStudents	
	SectionCode	AdClassSched.Section	
	ShiftId	AdClassSched.AdShiftId	
	StartDate	AdClassSched.StartDate	
	StudentSpecificMeetingSchedule	AdClassSched.AllowStudentSpecificMeeting	
	StudentSpecificMeetingScheduleDefaultMinutes	AdClassSched.DefaultMeetingLengthStudentSpecific	
	TermId	AdClassSchedTerm.AdTermId	
	WaitListMaximumSeats	AdClassSched.WaitListMaxnumOfSeats	
ClassSectionInstructor	Mapping is applicable to AdClassSchedInstructor only if instructor is secondary instructor. Primary instructor is stored in column on AdClassSched.		
	Id	AdClassSchedInstructor.AdClassSchedInstructorId	
	InstructorId	AdTeacher.SyStaffId	Join to AdTeacher on AdClassSchedInstructor.AdteacherId
	Type	N/A	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
Course			
	AddDropDays	AdCourse.AddDropDays	
	AddDropDaystype	AdCourse.AddDropCalendarDays	
	BusinessUnits	Collection of BusinessUnit	Join to SyCampusgroup on AdCourse.SyCampusGrpId and then to SyCampusList on SyCampusGrpId to retrieve the collection of SyCampusList.SyCampusIds that are associated to the instance of Course.
	Code	AdCourse.Code	
	CourseLevelId	AdCourse.AdCourseLevelId	
	CourseTypeId	AdCourse.AdCourseTypeId	
	CourseUnits	Collection of CourseUnit	
	CreatedByUserId	AdCourse.UserId	
	GradeLevel	AdCourse.GradeLevel	
	Id	AdCourse.AdCourseId	
	IsActive	AdCourse.Active	
	IsRemedialCourse	AdCourse.IsCourseRemedial	
	ModifiedByUserId	AdCourse.UserId	
	Name	AdCourse.Descrip	
	Note	AdCourse.Comments	
	PublishCode	AdCourse.CatalogCode	
CourseUnit			
	CourseId	AdCourse.AdCourseId	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	Id	N/A	Not sure how to populate this as there is nothing this maps to in existing Anthology Student schema. Purposely making this property nullable in contract because of this. Normally, Id property in contract is not nullable.
	Type	N/A	If Credits, then UnitValue is AdCourse.Credits. If Hours, then UnitValue is AdCourse.Hours.
	UnitValue	AdCourse.Hours, AdCourse.Credits	Value of Type dictates if Hours or Credits.
StudentAreasOfStudy	Current mapping logic only updates to the AdEnroll table. AdConcentrationbyEnrollment is not updated from this contract in the current implementation.		
	AreaOfStudyDetails	See AreasOfStudy class.	
	AreaofStudyId	AdEnroll.AdprogramVersionId	
	CatalogId	AdEnroll.AdCatalogYearId	
	DeclaredDate	AdEnroll.Startdate	
	Id	N/A	
	StudentEnrollmentPeriodId	AdEnroll.AdEnrollId	
	StudentId	AdEnroll.SyStudentId	
StudentCourse			
	ClassSectionId	AdEnrollSched.AdClassSchedId	
	ClassSectionSeatAllocationRuleId	N/A	
	CourseId	AdEnrollSched.AdCourseId	
	EndDate	AdEnrollSched.EndDate	
	ExpectedEndDate	AdEnrollSched.ExpectedEndDate	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	GradePoints	AdEnrollSched.Points	
	GradePostedDate	AdEnrollSched.DateGradePosted	
	GradeScaleId	AdEnrollSched.AdGradeScaleId	
	Id	AdEnrollSched.AdEnrollSchedId	
	IsAudit	AdEnrollSched.IsAudit	
	LastAttendanceDate	AdEnrollSched.LDA	
	LetterGrade	AdEnrollSched.AdGradeLetterCode	
	Note	AdEnrollSched.Comments	
	NumericGrade	AdEnrollSched.NumericGrade	
	PersonId	AdEnrollSched.SyStudentId (CONVERTED)	
	PreviousStatus	AdEnrollSched.PreviousStatus	
	StartDate	AdEnrollSched.StartDate	

<table>
<thead>
<tr><th>Anthology Class</th><th>Anthology Property</th><th>Anthology Student Table.Field Name</th><th>Comments</th></tr>
</thead>
<tbody>
<tr><td></td><td>Status</td><td>AdEnrollSched.Status</td><td>The mapping for Anthology Entity Status to Anthology Student Status is as follows:
<table>
<tr><th>Entity Status</th><th>Anthology Student Status</th></tr>
<tr><td>NotTaken</td><td>Future</td></tr>
<tr><td>Registered</td><td>Scheduled</td></tr>
<tr><td>CurrentlyAttending</td><td>Current</td></tr>
<tr><td>GradePosted</td><td>Complete</td></tr>
<tr><td>Withdrawal</td><td>Dropped</td></tr>
</table>
Since each Entity Status change can raise multiple events in Anthology Student, workflows using the Status property need to check for multiple status changes. Please refer to Check for StudentCourse.Status Changes for details.</td></tr>
<tr><td></td><td>StudentId</td><td>AdEnrollSched.SyStudentId</td><td></td></tr>
<tr><td></td><td>TermId</td><td>AdEnrollSched.AdtermId</td><td></td></tr>
<tr><td></td><td>TranscriptNote</td><td>AdEnrollSched.TranscriptComment</td><td></td></tr>
<tr><td></td><td>UnitValues</td><td>Collection of StudentCourseUnitValue</td><td></td></tr>
<tr><td colspan="3"><b>StudentCourseUnitValue</b></td><td></td></tr>
<tr><td></td><td>Id</td><td>N/A</td><td>Anthology Student does not have a separate units table. So this is ignored in mapping logic?</td></tr>
</tbody>
</table>

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	StudentCourseId	AdEnrollSched.AdEnrollSchedId	
	Type	Used to determine which AdEnrollSched fields to update	Valid Values for StudentCourseUnitValueType - "Credits" or "Hours"
	Units	AdEnrollSched.Credits, AdEnrollSched.Hours	If StudentCourseUnitValueType = "Credits" THEN AdEnrollSched.Credits ELSE AdEnrollSched.Hours
	UnitsAttempted	AdEnrollSched.CreditsAttempt, AdEnrollSched.HoursAttempt	If StudentCourseUnitValueType = "Credits" THEN AdEnrollSched.CreditsAttempt ELSE AdEnrollSched.HoursAttempt
	UnitsEarned	AdEnrollSched.CreditsEarned, AdEnrollSched.HoursEarned	If StudentCourseUnitValueType = "Credits" THEN AdEnrollSched.CreditsEarned ELSE AdEnrollSched.HoursEarned
StudentEnrollmentPeriod			
	AccountSummary	See AccountSummary on CMC.Nexus.Sis.StudentAccounts.	
	ApplicantTypeId	AdEnroll.AmApplicantTypeId	
	ApplicationReceivedDate	N/A	
	AreasOfStudy	Collection of StudentAreaOfStudy	
	AssignedAdmissionsRepId	AdEnroll.AmRepId	
	CampusId	AdEnroll.SyCampusId	
	EducationLevelId	AdEnroll.AmPrevEducId	
	EnrollDate	AdEnroll.EnrollDate	
	EnrollmentNumber	AdEnroll.StuNum	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	EnrollmentStatusId	AdEnroll.AdAttStatId	
	ExpectedGraduationDate	AdEnroll.GradDate	
	ExpectedStartDate	AdEnroll.ExpStartDate	
	ExternshipStartDate	AdEnroll.ExternBeginDate	
	GradeLevelId	AdEnroll.AdGradeLevelId	
	GraduationDate	AdEnroll.GradDate	
	Id	AdEnroll.AdenrollId	
	IpedsTransfer	AdEnroll.IPEDSTransfer	
	Lda	AdEnroll.LDA	
	MidpointDate	AdEnroll.MidDate	
	Note	AdEnroll.Comment	
	NsldsWithdrawalDate	AdEnroll.NSLDSWithdrawalDate	
	SapFlag	AdEnroll.Sap	
	StartDate	AdEnroll.StartDate	
	StartTermId	AdEnroll.AdtermId	
	StudentId	AdEnroll.SyStudentId	
	StudentStatusId	AdEnroll.SySchoolStatusId	
	TransferCredits	AdEnroll.TransferCredits	
Term			
	BusinessUnits	Collection of BusinessUnit	Join to SyCampusgroup on Adterm.SyCampusGrpId and then to SyCampusList on SyCampusGrpId to retrieve the collection of SyCampusList.SyCampusIds that are associated to the instance of Term.
	Code	AdTerm.Code	
	EndDate	AdTerm.EndDate	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	Id	Adterm.AdtermId	
	IsActive	AdTerm.Active	
	Name	AdTerm.Descrip	
	StartDate	AdTerm.StartDate	

Cmc.Nexus.Sis.Admissions

The following table shows the mapping of classes and properties in the Cmc.Nexus.Sis.Admissions entity to tables and fields in the Anthology Student database.

Cmc.Nexus.Sis.Admissions Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
Prospect	Mapping logic does not currently map to SyStudentInquiry.		
	AssignedAdmissionsRepId	SyStudent.AmRepId	
	AssignedStaffGroupId	N/A	
	AssociatedBusinessUnits	SyStudent.SyCampusId	Contract allows for multiple Business Unit values; however, only first value provided is mapped to Anthology Student.
	CreatedByUserId	SyStudent.UserId	
	DateAdded	SyStudent.DateAdded	
	DateModified	SyStudent.DateLstMod	
	EducationLevelId	SyStudent.AmPrevEducId	
	ExpectedStartDate	SyStudent.StartDate	
	HighSchoolGpa	SyStudent.HsAcademicGPA	
	Id	SyStudent.SyStudentId	
	LeadDate	SyStudent.LeadDate	
	LeadStatusId	SyStudent.SySchoolStatusId	
	LeadTypeId	SyStudent.AmLeadTypeId	
	Person	N/A	
	PrimaryLeadSourceId	SyStudent.AmLeadSrcId	
	RatingId	N/A	
	SecondaryLeadSource	N/A	
	Tasks	N/A	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	VendorOrganizationId	N/A	
ProspectLeadSource			
	Id	AmProspectLeadSrc.AmProspectLeadSrcId	
	LeadSourceId	AmprospectLeadSrc.AmLeadSrcId	

Cmc.Nexus.Sis.CareerServices

The following table shows the mapping of classes and properties in the Cmc.Nexus.Sis.CareerServices entity to tables and fields in the Anthology Student database.

Cmc.Nexus.Sis.CareerServices Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
StudentEmploymentHistory			
	EmployerId	PlStudentPlacement.PlEmployerId	
	Id	PlStudentPlacement.PlStudentPlacementId	
	PlacedDate	PlStudentPlacement.DatePlaced	
	Status	PlStudentPlacement.Status	
	StatusReasonId	N/A	
	StudentId	PlStudent.SyStudentId	
	StudentPlacementSummaryId	PlStudentPlacement.PlStudentId	
StudentPlacementSkill			
	Id	PlStudentSkill.PlStudentSkillId	
	SkillId	PlStudentSkill.PlSkillId	
	StudentPlacementSummaryId	PlStudentSkill.PlStudentId	
StudentPlacementSummary			
	Id	PlStudent.PlStudentId	
	PersonId	PlStudent.SyStudentId (CONVERTED)	
	PlacementStatusId	PlStudent.SySchoolStatusId	
	PlacementStatusReasonId	PlStudent.PlReasonId	
	StudentDegreeId	N/A	
	StudentEnrollmentPeriodId	PlStudent.AdEnrollId	
	StudentSkills	Collection of StudentPlacementSkill	

Cmc.Nexus.Sis.FinancialAid

The following table shows the mapping of classes and properties in the Cmc.Nexus.Sis.FinancialAid entity to tables and fields in the Anthology Student database.

Cmc.Nexus.Sis.FinancialAid Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
DirectLoanOrigination			
	BorrowerDefaultOnLoans	FaLoan.BorrowerDefaultOnLoans	
	PromNoteBorrowerSigned	FaLoan.PnSignedByBorrower	
	DisclosureStatementPrintCode	FaLoan.PrintDisclosureCode	
	IncludeOnManifest	FaLoan.PnIncludeOnManifest	
	InterestRebatePercentage	FaLoan.InterestRebatePct	
	ManifestDate	FaLoan.ManifestDate	
	MpnExpirationDate	FaLoan.MpnExpirationDate	
	MpnIdentifier	FaLoan.PnDirectLoanId?	
	MpnLinkIndicator	FaLoan.MpnIndicator	
	MpnType	FaLoan.MpnType	
	OriginationAcknowledgeDate	FaLoan.AcknowledgeDate	
	OriginationBatchIdentifier	FaLoan.OriginationBatchId	
	OriginationDate	FaLoan.OriginationDate	
	OriginationFeePercentage	FaLoan.OriginationFeePct	
	OriginationRejectCodes	FaLoan.OrigRejectCodes	
	OriginationStatus	FaLoan.OriginationStatus	
	PlusLoan	See DirectLoanOriginationPlus class.	
	PreparatoryProfessionalCourseWork	FaLoan.PreProfessionalCourseWorkIndicator	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	PromNoteAcceptedAmountDate	FaLoan.PnAcceptedAmountDate	
	PromNotePrintCode	FaLoan.PnPrintIndicator	
	PromNotePrintedDate	FaLoan.DatePnPrinted	
	PromNoteSignedDate	FaLoan.DatePnSigned	
	PromNoteSignedReceivedDate	FaLoan.DateSignedPnReceived	
	PromNoteStatus	FaLoan.PnStatus	
	RebateAmount		
	UnsubLoan	See DirectLoanOriginationUnsub class.	
DirectLoanOriginationPlus			Inherits from DirectLoanOrigination.
	CreditDecisionDate	FaLoan.PlusCreditDecisionDate	
	CreditDecisionStatus	FaLoan.PlusCreditDecisionStatus	
	StudentCitizenStatus	FaLoan.StudentCitizenStatus	
	StudentDefaultOnLoans	FaLoan.StudentDefaultOnLoans	
DirectLoanOriginationSub			Inherits from DirectLoanOrigination. No unique properties in this class other than what is inherited from DirectLoanOrigination.
DirectLoanOriginationUnsub			Inherits from DirectLoanOrigination.
	AdditionalUnsubEligibility	FaLoan.UnsubEligibilityFlag	
	ParentDeniedPlusLoan	FaLoan.ParentRejectedForPlus	
DirectLoanScheduledDisbursement			Inherits from StudentAwardScheduledDisbursement.
	ActualDisbursementDate	FaSched.ActDisbDate	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	CodStatus	FaSched.DlStatus	
	DisbursementPercentage	FaSched.DlPercentage	
	OverrideDisbursementDate	FaSched.OverrideDisbDate	
	RebateAmount	FaSched.InterestRebateAmt	
FundSource			
	Code	FaFundSource.Code	
	FundSourceType	FaFundSource.Type	
	Id	FaFundSource.FaFundSourceId	
	Name	FaFundSource.Descrip	
	TitleIv	FaFundSource.TitleIv	
PaidDisbursement			
	AmountPaid	FaDisb.ActualAmount	
	CheckNumber	FaDisb.CheckNumber	
	Id	FaDisb.FaDisbId	
	Note	N/A	
	PaidDate	FaDisb.DateDisb	
	Status	FaDisb.Status	
	StudentAcademicYearPaymentPeriod	FaDisb.AdtermId, FaDisb.FaStudentAyPaymentPeriodId	Depending on what type of payment period is being associated in Anthology Student, this property may map to multiple Anthology Student fields.
PellScheduledDisbursement			Inherits from StudentAwardScheduledDisbursement.
	ClockHours	FaSched.ClockHours	
	CodStatus	FaSched.PgDisbStatus	
	EnrollmentStatus	FaSched.EnrollmentStatus	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	PaymentPeriodBeginDate	FaSched.PaymentPeriodBeginDate	
Refund			
	Amount	FaRefund.Amount	
	CheckNumber	FaRefund.CheckNo	
	DueDate	FaRefund.DateDue	
	Id	FaRefund.FaRefundId	
	Note	FaRefund.Comment	
	PaidDate	FaRefund.DateSent	
	ReturnMethod	FaRefund.ReturnMethod	
	Status	FaRefund.Status	
	StudentAcademicYearPaymentPeriod	FaRefund.AdTermId, FaRefund.FaPmtPeriodId, FaRefund.FaStudentAyPaymentPeriodId	Depending on what type of payment period is being associated in Anthology Student, this property may map to multiple Anthology Student fields.
ScheduledDisbursement			
	AmountExpected	FaSched.NetAmount	
	ExpectedDate	FaSched.DateSched	
	Id	FaSched.FaSchedId	
	Note	N/A	
	Status	FaSched.Status	
	StudentAcademicYearPaymentPeriod	FaSched.AdTermId, FaSched.FaPmtPeriodId, FaSched.FaStudentAyPaymentPeriodId, FaSched.FaStudentLpPaymentPeriodId	Depending on what type of payment period is being associated in Anthology Student, this property may map to multiple Anthology Student fields.
StudentAcademicYear			
	AcademicYearMonths	FaStudentAy.MonthsInAy	
	AcademicYearSequence	FaStudentAy.Sequence	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	AcademicYearTemplateId	FaStudentAy.FaAcademicYearId	
	AcademicYearUnits	FaStudentAy.CreditHoursInAy	
	AcademicYearWeeks	FaStudentAy.WeeksInAy	
	AwardYears	FaStudentAy.AwardYear1, FaStudentAy.AwardYear2	
	BudgetId	FaStudentAy.FaBudgetId	
	BudgetItems	Collection of StudentAcademicYearBudgetItem	Changes to Budget Items will trigger a separate event from changes to main Student Academic Year event entity. For example, if you change Housing from Off Campus to On Campus, you will receive two events – one for housing status change on the main entity and one for changes to the Budget Items collection for Room and Board.
	CreatedByUserId	FaStudentAy.UserId	
	EligibleHealthProfession	FaStudentAy.HPPALevel	
	EndDate	FaStudentAy.EndDate	
	FaAdvisorId	SyAdvisorByEnroll.SyStaffId	Find row in SyAdvisorByEnroll WHERE AdEnrollId = StudentEnrollmentPeriodId AND AdvisorModule = 'FA'
	FirstTimeBorrower	FaStudentAy.FirstTimeBorrower	
	GradeLevelId	FaStudentAy.AdGradeLevelId	
	HousingStatus	FaStudentAy.HousingStatusCode	
	Id	FaStudentAy.FaStudentAyId	
	ModifiedByUserId	FaStudentAy.UserId	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	Note	Most recent comment is stored in FaStudentAy.Comment. Only for Saving event. N/A for Saved event.	Comments are now stored in separate table FaStudentAy.Comment and not in the FaStudentAy.AyComments field. Additional comments can be found in that table, but are not provided within the event.
	PersonId	SyStudent.SyStudentId (CONVERTED)	
	PlusCreditDecision	FaStudentAy.PlusCreditDecisionStatus	
	StartDate	FaStudentAy.StartDate	
	StudentAwardSummaries	Collection of StudentAwardSummary	
	StudentEnrollmentPeriodId	FaStudentAy.AdEnrollId	
	UnitsExpectedToComplete	FaStudentAy.CreditHoursExpToComplete	
	WeeksEnrolled	FaStudentAy.WeeksEnrolledInAy	
	WeeksNonEnrolled	FaStudentAy.WeeksNonEnroll	
StudentAcademicYearBudgetItem			
	Amount	FaStudentAy.Tuition, BooksSupplies, InstitutionalCharges, RoomBoard, Travel, OtherAmount#	Depending on what type of budget cost item it is will determine which field in FaStudentAy this is mapped to.
	CostDescription	N/A	Description of budget cost item. Not mapped to anything in Anthology Student.
	CostTypeChargeCodeId	N/A	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	CostType	Not mapped. Is Enum property.	Not mapped to field in Anthology Student. However, this will determine which type of budget cost item this is, which will determine which fields in the FaStudentAy record to map to. The valid values for this property in Anthology are: Tuition, Books/Supplies, Room/Board, Travel, Bank Fees, Other
	IsInstitutionalCharge	FaStudentAy.OtherInst#	
	StudentAcademicYearId	FaStudentAy.FaStudentAyId	
StudentAcademicYearPaymentPeriod			This does not map directly to any table in Anthology Student. Depending on how payment periods are defined, this Anthology class may map to AdTerm, FaStudentAyPaymentPeriod, or FaStudentAyLPPaymentPeriod.
	Id		
	PaymentPeriod		
	PaymentPeriodEndDate		
	PaymentPeriodStartDate		
	Sequence		
	StudentAcademicYearId		
StudentAward			
	AwardAmount	FaStudentAid.AmountPackaged	
	CreateDate	FaStudentAid.DateAdded	
	CreatedByUserId	FaStudentAid.UserId	
	FundSourceId	FaStudentAid.FaFundSourceId	
	Id	FaStudentAid.FaStudentAidId	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	ModifiedbyUserId	FaStudentAid.UserId	
	Note	FaStudentAid.Comment	
	PaidDisbursements	Collection of StudentAwardPaidDisbursement	This entity collection is not mapped in Saved or Saving events.
	Refunds	Collection of Refund.	This entity collection is not mapped in Saved or Saving events.
	ScheduledDisbursements	Collection of StudentAwardScheduledDisbursement	This entity collection is available only in Saving events, not in Saved events.
	ScheduledDisbursementsTemplateId	N/A	
	Status	FaStudentAid.Status	
	StudentAcademicYearId	FaStudentAid.FaStudentAyId	
	StudentAwardSummaryId	N/A	
StudentAwardGrant			Inherits from StudentAward. No other properties in this class other than what is inherited from StudentAward.
StudentAwardLoan			Inherits from StudentAward.
	DirectLoanDetail	See DirectLoanOrigination class.	This entity is available only in Saving events, not in Saved events.
	GuarantorId	FaStudentAid.FaGuarantorId	
	LenderFee	FaStudentAid.BankFee	
	LenderId	FaStudentAid.FaBankId	
	LoanIdentifier	FaLoan.CommonlineLoanId or FaLoan.DirectLoanId or N/A	If a Direct loan, then FaLoan.DirectLoanId. If a loan that will be processed via Commonline, then FaLoan.CommonlineLoanId.
	LoanPeriodEndDate	FaStudentAid.DateLoanStart	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	LoanPeriodStartDate	FaStudentAid.DateLoanEnd	
	ServicerId	FaStudentAid.FaServicerId	
StudentAwardPaidDisbursement			
	DepositDate	FaDisb.DateDeposited	
	DisbursementNumber	FaDisb.DisbNum	
	SignedDate	FaDisb.DateSigned	
StudentAwardPell			Inherits from StudentAward.
	AcademicCalendar	FaStudentPell.AcademicCalendar	
	AdministrativeRelief	FaStudentPell.AdminRelief	
	EnrollmentDate	FaStudentPell.EnrollDate	
	EnrollmentStatus	FaStudentPell.PellEnrollmentStatus	
	IncarceratedCode	FaStudentPell.IncarceratedCode	
	LifetimePercentageEligibilityUsed	FaStudentPell.LifetimeEligUsed	
	NumberOfPaymentPeriodsInAcademicYear	FaStudentPell.NumPayPeriods	
	OriginationAmount	FaStudentPell.PellAmount	
	OriginationStatus	FaStudentPell.OriginationStatus	
	PaymentMethodology	FaStudentPell.PaymentMethodology	
	PercentageEligibilityUsed	FaStudentPell.TotalEligibilityUsed	
StudentAwardScheduledDisbursement			Inherits from ScheduledDisbursement.

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	DirectLoanDisbursement	See DirectLoanScheduledDisbursement class.	
	DisbursementNumber	FaSched.Disbnum	
	LenderFee	FaSched.BankFee	
StudentAwardSummary			
	AwardDate	FaStudentAy.PackageDate	
	AwardedEnrollmentStatusId	FaStudentAy.PackagedToAdAttStatId	
	AwardingStatusId	FaStudentAy.FaPackStatusId or FaStudentAy. AwardYear2FaPackStatusId	If AwardYear is in FaStudentAy.AwardYear2, then the awarding status will be in the AwardYear2FaPackStatusId attribute.
	AwardMethodId	FaStudentAy.FaPackMethId	
	AwardNoticePrinted	FaStudentAy.AwardNoticePrinted	
	AwardNoticeSigned	FaStudentAy.AwardNoticeSigned	
	AwardRevised	FaStudentAy.PackageRevised	
	AwardRevisedNoticePrinted	FaStudentAy.RevisedNoticePrinted	
	AwardRevisedNoticeSigned	FaStudentAy.RevisedNoticeSigned	
	AwardYear	FaStudentAy.AwardYear1 or FaStudentAy.AwardYear2	
	AwardYearId	FaYear.FaYearId	
	Id	FaStudentAy.FaStudentAyId	
	IsModelOverride	FaStudentPell.ModelOverride	
	Model	FaStudentPell.Model	
	PersonId	SyStudent.SyStudentId (CONVERTED)	

Cmc.Nexus.Sis.StudentAccounts

The following table shows the mapping of classes and properties in the Cmc.Nexus.Sis.StudentAccounts entity to tables and fields in the Anthology Student database.

Cmc.Nexus.Sis.StudentAccounts Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
AccountChargeTransaction			
	ChargeCodeId	SaTrans.SaBillCode	Contract property is Id; however, billcode in SaTrans is char. So, Id is retrieved from SaBillCode.
	InvoiceNumber	SaTrans.Ref	
AccountSummary			
	AccountBalance	AdEnroll.ArBalance	
	AccountStatuses	Collection of AccountStatusDetail	Multiple Account Statuses are allowed. Currently, in Anthology Student each different account status specified for an enrollment is an instance in SaEnrollAcctStatus.
	BillingMethodId	AdEnroll.SabillingMethodId	
	BusinessUnitId	N/A	
	Id	AdEnroll.AdEnrollId	
	PersonId	SyStudent.SyStudentId (CONVERTED)	
	StudentEnrollmentPeriodId	AdEnroll.AdEnrollId	
AccountTransaction			
	AddUserId	SaTrans.AddUserId	
	AccountChargeTransaction - Derived Type	See AccountChargeTransaction class.	
	AccountPaymentTransaction - Derived Type	See AccountPaymentTransaction class.	
	Amount	SaTrans.Amount	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	BillingPeriodId	SaTrans.AdtermId	
	BusinessUnitId	SaTrans.SyCampusId	
	Description	SaTrans.Descrip	
	Id	SaTrans.SaTransId	
	PersonId	SaTrans.SyStudentId (CONVERTED)	
	PostDate	SaTrans.PostDate	
	ProspectId	SaTrans.SyStudentId	
	Reference	SaTrans.Ref	
	StudentBillingPeriodId	SaTrans.FaPmtPeriodId	
	StudentEnrollmentPeriodId	SaTrans.AdEnrollId	
	TransactionDate	SaTrans.Date	
	TransactionType	SaTrans.Type	
AccountPaymentTransaction			
	CheckNumber	SaTrans.CheckNo	
	ReceiptNumber	SaTrans.ReceiptNo	
AccountStatusDetail			
	AccountStatusId	SaCollectionAccountStatus. SaAcctStatusID	For a CollectionAccount event, the fields listed in the column to the left are returned . For an AccountSummary event, the following fields are returned: • SaEnrollAcctStatus.SaAcctStatusID • SaEnrollAcctStatus.SaEnrollAcctStatusID
	Id	SaCollectionAccountStatus. SaCollectionAccountStatusID	
CollectionAccount			

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	AccountStatuses	Collection of AccountStatusDetail	Multiple Account Statuses are allowed. Currently, in Anthology Student each different account status specified for a collection account is an instance in SaCollectionAccountStatus.
	BlockStatement	SaCollections.BlockStatement	
	DunningProcessOff	SaCollections.DunningProcessoff	
	Id	SaCollections.SaCollectionsId	
	LastStatementAmount	SaCollections.StatementAmount	
	LastStatementDate	SaCollections.StatementDate	
	PaymentPlanSummaryId	SaCollections.FaStudentAidId	
	PersonId	SaCollections.SyStudentId (CONVERTED)	
	ProspectId	SaCollections.SyStudentId	
	ReadyForCollectionDate	SaCollections.ReadyForCollectionDate	
	StatementNote	SaCollections.StatementMemo	
	StudentEnrollmentPeriodI	SaCollections.AdEnrollId	
StudentPaymentPlan			
	FirstPaymentDate	FaStudentAid.FirstPayDate	
	FundSourceId	FaStudentAid.FaFundSourceId	
	Id	FaStudentAid.FaStudentAidId	
	InterestOnlyUntilDate	FaStudentAid.IntOnlyuntilDate	
	InterestRate	FaStudentAid.InterestRate	
	Note	FaStudentAid.Comment	
	NumberOfPayments	FaStudentAid.NumberPayments	

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	PaidPayments	Collection of StudentPaymentPlanPayment	This entity collection is not mapped in Saved or Saving events.
	PaymentAmount	FaStudentAid.PaymentAmount	
	PaymentFrequency	N/A	This appears to just be a field on the UI in Anthology Student and is not persisted in the database.
	PaymentFrequencyDays	FaStudentAid.PaymentFreq	
	PrincipalAmount	FaStudentAid.PrincipalBalance	
	Refunds	Collection of Refund (see Cmc.Nexus.Sis.FinancialAid)	This entity collection is not mapped in Saved or Saving events.
	ScheduledPayments	Collection of StudentPaymentPlanScheduledPayment	
	SecondaryInterestRate	FaStudentAid.SecondaryInterestRate	
	SecondaryInterestRateEffectiveDate	FaStudentAid.SecondaryInterestRateEffDate	
	StatementAddressId	FaStudentAid.SyAddressId	
	Status	FaStudentAid.Status	
	StudentAcademicYearId	FaStudentAid.FaStudentAyId	
StudentPaymentPlanPayment			
	InterestAmount	FaDisb.InterestAmount	
	PrincipalAmount	FaDisb.ActualAmount - FaDisb.InterestAmount	Field for PrincipalAmount does not exist in FaDisb schema. Amount is calculated by subtracting InterestAmount from ActualAmount.
StudentPaymentPlanScheduledPayment			
	InterestAmount	FaSched.InterestAmount	
	PrincipalAmount	FaSched.PrincipalAmount	
StudentPaymentPlanStatement			

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	AmountDue	SaStatementHistory.StatementAmtDue	
	AmountPastDue	N/A	
	ClosingDate	SaStatementHistory.ClosingDate	
	GeneratedDate	SaStatementHistory.DatePrinted	
	Id	SaStatementHistory.SaStatementHistoryId	
	StudentPaymentPlanId	SaStatementHistory.FaStudentAidId	

Cmc.Nexus.StudentServices

The following table shows the mapping of classes and properties in the Cmc.Nexus.StudentServices entity to tables and fields in the Anthology Student database.

Cmc.Nexus.StudentServices Mapping

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
DisabilityType			
	Id	SsDisabilityType.SsDisabilityTypeId	
StudentAthleticDetail			
	AthleticStatusId	SsAthleticDetail.SsAthleticStatusId	
	Id	SsAthleticDetail.SsAthleticDetailId	
	LastActiveTermId	SsAthleticDetail.AdTermId	
	RecruitmentTypeId	SsAthleticDetail.SsRecruitmentTypeId	
	RemainingEligibility	SsAthleticDetail.RemainingEligibility	
	SportId	SsAthleticDetail.SsSportsId	
	StudentId	SsAthleticDetail.SyStudentId	
StudentDisabilityDetail			
	DisabilityStatusId	SsStudentDisabilityDetail.SsDisabilityStatusId	
	DisabilityType	SsStudentDisabilityDetail.SsDisabilityTypeIds	Collection of DisabilityType
	Id	SsStudentDisabilityDetail.SsStudentDisabilityDetailId	
	IsDisabled	SsStudentDisabilityDetail.Disabled	
	IsPriorityRegistration	SsStudentDisabilityDetail.PriorityRegistration	
	IsRegistrationAssistanceNeeded	SsStudentDisabilityDetail.RegistrationAssistance	
	Note	SsStudentDisabilityDetail.Comments	
	StudentId	SsStudentDisabilityDetail.SyStudentId	
StudentVeteranDetail			

Anthology Class	Anthology Property	Anthology Student Table.Field Name	Comments
	BenefitsReceived	SsStudentVeteranDetail. SsveteranBenefitIds	Collection of VeteranBenefit
	Id	SsStudentVeteranDetail. SsStudentVeteranDetailId	
	LastcertifiedTermId	SsStudentVeteranDetail.AdtermId	
	StudentId	SsStudentveteranDetail.SyStudentId	
	VeteranCertificationTypeId	SsStudentVeteranDetail. SsVeteranCertificationTypeId	
	VeteranTypes	SsStudentVeteranDetail. SsVeteranCodeIds	Collection of VeteranType
VeteranBenefit			
	Id	SsVeteranBenefit.SsVeteranBenefitId	
VeteranType			
	Id	SsVeteranCode.SsVeteranCodeId	

Events

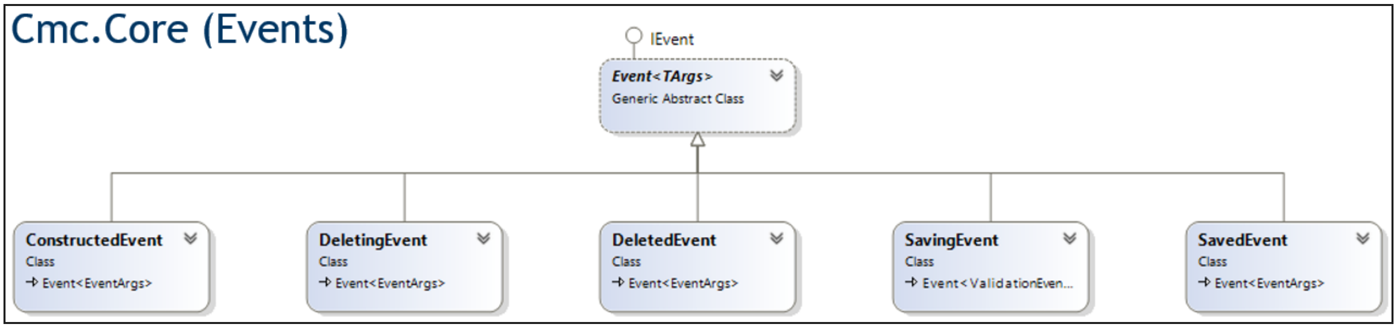
Events thats are captured in Anthology can be used to trigger workflow activities.

Events Overview

The Event Broker listens for incoming events from clients, determines the name of the event, forwards the event to the configured event handler, and, if required, returns a response to the event. Event messages contain enough basic information to be handled without the need to retrieve additional data from APIs.

The events that are exposed to the Event Broker can be consumed in custom code (for example, C# event handlers) or workflows that automate tasks and enable data to be exchanged between systems.

Anthology events are grouped in the categories depicted below.



- **Saving events** and **Deleting events** are captured and visible at the UI level. VB .NET code is required to intercept these events. Data validation occurs. Saving and Deleting event workflows must be stored on the host that is running the application on which the event is captured, for example, Anthology Student.
- **Saved events** and **Deleted events** are captured at the database trigger level. These events are only visible in the event log of the Windows Service NextGen Nexus Event Workflows. Saved and Deleted event workflows must be stored on a host that has a direct database connection, for example, COM Server.
- **Constructed events** are captured and visible at the UI level when the components of a record are assembled. No data validation occurs. VB .NET code is required to intercept these events.

The available event categories depend on the entities. For example, the Person entity in Anthology CRM is associated with Constructed, Saving, and Saved events, while the Student Enrollment Period entity in Anthology Student is associated only with Saved and Deleted events.

Forms Builder events fall into a different category. These events are triggered whenever the **Raise Event** rule is encountered in a sequence.

Note: The initial Workflow and Eventing versions support Saved events and Saving events for CampusNexus CRM and Anthology Student and the Raise Event rule for Forms Builder.

Workflow Composer enables you to intercept the events and create activities that are triggered by the events. Activities in a workflow can be triggered by Saved and Saving events.

Events published to the Event Broker are application-specific, that is, a distinct set of events is available for Anthology Student (see SIS Events), another set of events is available for CampusNexus CRM, another set for Forms Builder, and so on. Contracts define the messages that will be exchanged between the applications.

Cmc.Core Events

The following events are common to all workflows regardless of the application, for example Anthology Student, CampusNexus CRM, or Forms Builder.

Cmc.Core Events

Event	Property	Description

Time-based event (e.g., duration, schedule)	`Entity: Schedule`
	`Event: ScheduleOccurrence`
		This event enables you to create non-request activated worfklows, that is, workflows based on time or events that occur outside of IIS. The SQL Server Agent Job scheduling is used to trigger workflows based on time.
		An example of a time-based event is a Delay activity. Workflows with a Delay activity can be explicitly paused, unloaded, and resumed by using persistence. For more information about workflow persistence, see http://msdn.microsoft.com/en-us/library/dd489420(v=vs.110).aspx.

SIS Events

The following events are specific to Anthology Student.

- Saving events are triggered just prior to data in an Anthology Student form being saved to the database.
- Saved events are triggered just after data is saved to the database.

SIS Saving Events

Saving events are triggered just prior to data in a form being saved to database and are most often used to CreateValidationItems on a form. You can configure Error, Information, or Warning messages that are displayed if any of the data entered on the form fails the configured validation rules. Event are captured and visible at the UI level. VB .NET code is required to intercept these events. Saving event workflows must be stored on the host that is running the application on which the event is captured, for example, Anthology Student. The workflow Check Approved Grants for Comments is an example of a workflow for a saving event. You can create any combination of workflow activities to formulate custom business rules that the system uses to ensure that quality data is being entered. You can use workflows to:

- Set, change, or remove values for specific fields.
- Perform validation on one or more fields.
- Trigger additional activities to be performed based on event data.

SIS Saving Events

Contracts	Anthology Student Form	Entity Mapping
Student Master Form		
Cmc.Nexus.Contracts > Cmc.Nexus > Person	Student > Student Master (frmAmStudMaster or frmAMStudMasterShort)	Cmc.Nexus • Person
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Student Master form. Example: A workflow assigns an email address to a student. When the Status field is changed from New Lead to Interviewed, the student's primary email address is moved to the Other email field and the primary email field is populated with a new email address that is created using the first three letters of `FirstName` and the first five letters of `LastName` followed by `@myschool.edu`.		
Student Enrollment Wizard		
Cmc.Nexus.Contracts > Cmc.Nexus > Person	Daily > Admissions > Enroll Student (frmAmEnroll) or View > Academic Records > Enrollment (frmAmEnroll)	Cmc.Nexus • Person Cmc.Nexus.Sis.Academics • StudentEnrollment Periods
This event enables you to create workflow activities that are triggered when the Next button is clicked on any step (page) of the Student Enrollment wizard and when the Finish button is clicked. You can create workflows that are triggered at specific points in the enrollment process. Example: A workflow checks the student's address when an attempt is made to enroll the student in a program that is not approved in the state where the student lives. The workflow prevents the enrollment if the student's address is not in a state where the program is approved.		
Financial Aid Academic Year Form		

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Academic Year	Student > Financial Aid > Academic Year (frmFaStudentAY)	Cmc.Nexus.Sis.FinancialAid • StudentAcademicYear
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Academic Year form.		
Financial Aid Loan Form		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Loan Detail	Student > Financial Aid > Packaging > Add > New Source of Aid > Loan (frmFaStudLoan)	Cmc.Nexus.Sis.FinancialAid • StudentAward • StudentAwardLoan • StudentAwardSummary
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Loan form. Example: Your institution requires students to complete an online course on financial responsibility if they request loans of more than $2,500 per academic year. You create a workflow that checks the gross loan amount and alerts the user when the amount is greater than $2,500 for an academic year so that the loan is not packaged prior to the completion of the online course.		
Financial Aid Grant / Scholarship Form		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Grant Detail	Student > Financial Aid > Packaging > Add > New Source of Aid > Grant (frmFaStudGrant)	Cmc.Nexus.Sis.FinancialAid • StudentAward • StudentAwardSummary
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Grant / Scholarship form. Example: Your institution requires approvers to add comments when they approve a grant / scholarship for a student. You create a workflow that checks for entries in the Comments field when the form is saved with a status of 'Approved'. See workflow example Check Approved Grants for Comments.		
Financial Aid Grant / Scholarship Form (Source = Pell)		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Award Pell	Student > Financial Aid > Packaging > Add > New Source of Aid > Grant, Source = Pell (frmFaStudPell)	Cmc.Nexus.Sis.FinancialAid • StudentAward • StudentAwardSummary
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Grant / Scholarship form with Source selection of 'Pell'. Example: Your institution requires approvers to add comments when they approve a Pell grant for a student. You create a workflow that checks for entries in the Comments field when the form is saved with a status of 'Approved'.		
Financial Aid Cash Payment/Other Form		

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Student Payment Plan	Student > Financial Aid > Packaging > Add > New Source of Aid > Student Payment/Other (frmFaStudCashOther)	Cmc.Nexus.Sis.StudentAccounts • StudentPaymentPlan
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Cash Payment/Other form. Example: Your institution requires multiple payments if the cash amounts is above $800. You create a workflow that validates the number of payments when the form is saved with a cash amount above $800.		
Post Charges Form		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Account Charge Transaction	Daily > Student Accounts > Post Charges OR Student Accounts > Ledger Cards > Post Charges (frmSaTransTrxs)	Cmc.Nexus.Sis.StudentAccounts • AccountChargeTransaction
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Post Charges form. Example: The Academic Year, Term, and Payment Period fields are not required fields on the Post Charges form in Anthology Student, but your institution requires these fields to be populated when charges are posted. You create a validation workflow that checks whether the user specified the Academic Year, Term, and Payment Period. If these fields are not populated, an error message is displayed and the user cannot save the transaction or adjustment. See Context Property for hints about how to determine the type of event (PostCharge or AdjustCharge).		
Class Scheduling Form		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Class Section	Daily > Academic Records > Schedule Classes (New/Edit) (frmAdClassSchedOne)	Cmc.Nexus.Sis.Academics • ClassSection

Contracts	Anthology Student Form	Entity Mapping
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Class Schedule form. Proper course section configuration is important for the SIS to function properly. Decisions made on this form such as attendance type, delivery method, and shift can have a ripple affect in the SIS application if they are not set properly. Outside of the required fields, workflows can ensure that courses are configured properly. Examples: • A workflow checks that the section number starts with the current year of the start date, e.g., 2015SPRING-01, or checks if the section number already exists, otherwise an information message is displayed. • A workflow checks that enrollment status credits do not exceed course credits, otherwise an error message is displayed. • A workflow checks that courses with a delivery type of 'Online' have the code '-O' at the end of the section number, otherwise an error message is displayed. • A workflow checks that courses are only scheduled to start on a Tuesday, otherwise a warning message is displayed.		
Transaction Adjustment Form		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Account Charge Transaction	View > Student Accounts > Ledger Cards > Adjust Transaction OR Student Accounts > Ledger Cards > Adjust Transaction (frmSaLedgerAdjustment)	Cmc.Nexus.Sis.StudentAccounts • AccountChargeTransaction
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Transaction Adjustment form. Workflow activities can help to ensure the posting of accurate charges and adjustments on student accounts. Workflows compensate for the fact that the data dictionary in Anthology Student does not always allow administrators to set required fields or validate the data of required fields before saving a form. Example: The Academic Year and Term are not required fields on the Transaction Adjustment form in Anthology Student, but your institution requires these fields to be populated when charges are posted. You create a validation workflow that checks whether the user specified the Academic Year and Term. If these fields are not populated, an error message is displayed, and the user cannot save the transaction adjustment. See Context Property for hints about how to determine the type of event (PostCharge or AdjustCharge).		
Courses Code Setup Form		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Course	Lists > Academic Records > Courses > Add/Edit > Courses Code Setup (frmAdCourse)	Cmc.Nexus.Sis.Academics • Course

<table>
<tr><th>Contracts</th><th>Anthology Student Form</th><th>Entity Mapping</th></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Courses Code Setup form.
Example:
A workflow checks if the PublishCode matches the Course Code. When the codes don't match, a custom validation message similar to the following appears in Anthology Student: <i>INFORMATION: The PublishCode does NOT match the Course Code - Students will only see the PublishCode on their transcripts.</i></td></tr>
<tr><td colspan="3">Address Form</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus > Person Address</td><td>Contact Manager > Addresses (frmSyStudAddresses)</td><td>Cmc.Nexus
• PersonAddress</td></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Address form. The Entity is the PersonAddress.
Example:
A workflow provides custom validation messages on the fields of the Address form, e.g., Address Type, Title, Last Name, First Name, Seasonal Dates, Effective Dates, so that Contact Manager activities always use correct address information.</td></tr>
<tr><td colspan="3">Athletics Form</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentService > Student Athletic Detail</td><td>View > Student Services >Athletics (frmSsAthletics)</td><td>Cmc.Nexus.StudentServices
• StudentAthleticDetail</td></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Athletics form.
<b>Note</b>: The Context property for this event is "StudentAthleticDetail Saving Com". For more information, see Context Property.</td></tr>
<tr><td colspan="3">Disability Services Form</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentService > Student Disability Detail</td><td>View > Student Services > Disabilities (FrmSsDisabilityService)</td><td>Cmc.Nexus.StudentServices
• StudentDisabilityDetail</td></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Disability Service form.
<b>Note</b>: The Context property for this event is "StudentDisabilityDetail Saving Com". For more information, see Context Property.</td></tr>
<tr><td colspan="3">Veteran Information Form</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentService > Student Veteran Detail</td><td>View > Student Services > Veteran Information(FrmSsVeteranInformation)</td><td>Cmc.Nexus.StudentServices
• StudentVeteranDetail</td></tr>
</table>

Contracts	Anthology Student Form	Entity Mapping
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Veteran Information form. **Note**: The Context property for this event is "StudentVeteranDetail Saving Com". For more information, see Context Property.		
Anthology Contract	**Anthology Student Form**	**Entity Mapping**
Student Master Form		
Cmc.Nexus.Contracts > Cmc.Nexus > Person	Student > Student Master (frmAMStudMasterShort)	Cmc.Nexus • Person
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Student Master form. Example: A workflow assigns an email address to a student. When the Status field is changed from New Lead to Interviewed, the student's primary email address is moved to the Other email field and the primary email field is populated with a new email address that is created using the first three letters of `FirstName` and the first five letters of `LastName` followed by `@myschool.edu`.		
Student Enrollment Wizard		
Cmc.Nexus.Contracts > Cmc.Nexus > Person	Daily > Admissions > Enroll Student (frmAmEnroll)	Cmc.Nexus • Person Entity: Cms.Nexus.Sis • StudentEnrollmentPeriods Cmc.Nexus.Sis.Academics • StudentEnrollmentPeriods
This event enables you to create workflow activities that are triggered when the Next button is clicked on any step (page) of the Student Enrollment wizard and when the Finish button is clicked. You can create workflows that are triggered at specific points in the enrollment process. Example: A workflow checks the student's address when an attempt is made to enroll the student in a program that is not approved in the state where the student lives. The workflow prevents the enrollment if the student's address is not in a state where the program is approved.		
Financial Aid Academic Year Form		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Academic Year	Student > Financial Aid > Academic Year (frmFaStudentAY)	Cmc.Nexus.Sis.FinancialAid • StudentAcademicYear
This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Academic Year form.		
Financial Aid Loan Form		

<table>
<tr><th>Contracts</th><th>Anthology Student Form</th><th>Entity Mapping</th></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Loan Detail</td><td>Student > Financial Aid > Packaging > Add > New Source of Aid > Loan (frmFaStudLoan)</td><td>Cmc.Nexus.Sis.FinancialAid
• StudentAward
• StudentAwardLoan
• StudentAwardSummary</td></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Loan form.

Example: Your institution requires students to complete an online course on financial responsibility if they request loans of more than $2,500 per academic year. You create a workflow that checks the gross loan amount and alerts the user when the amount is greater than $2,500 for an academic year so that the loan is not packaged prior to the completion of the online course.</td></tr>
<tr><td colspan="3">Financial Aid Grant / Scholarship Form</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Grant Detail</td><td>Student > Financial Aid > Packaging > Add > New Source of Aid > Grant (frmFaStudGrant)</td><td>Entity: Cmc.Nexus.Sis.FinancialAid
• StudentAward
• StudentAwardSummary</td></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Grant / Scholarship form.

Example: Your institution requires approvers to add comments when they approve a grant / scholarship for a student. You create a workflow that checks for entries in the Comments field when the form is saved with a status of 'Approved'. See workflow example Check Approved Grants for Comments.</td></tr>
<tr><td colspan="3">Financial Aid Grant / Scholarship Form (Source = Pell)</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Award Pell</td><td>Student > Financial Aid > Packaging > Add > New Source of Aid > Grant, Source = Pell (frmFaStudPell)</td><td>Entity: Cmc.Nexus.Sis.FinancialAid
• StudentAward
• StudentAwardSummary</td></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Grant / Scholarship form with Source selection of 'Pell'.

Example: Your institution requires approvers to add comments when they approve a Pell grant for a student. You create a workflow that checks for entries in the Comments field when the form is saved with a status of 'Approved'.</td></tr>
<tr><td colspan="3">Financial Aid Cash Payment/Other Form</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Student Payment Plan</td><td>Student > Financial Aid > Packaging > Add > New Source of Aid > Student Payment/Other (frmFaStudCashOther)</td><td>Cmc.Nexus.Sis.StudentAccounts
• StudentPaymentPlan</td></tr>
<tr><td colspan="3">This event enables you to create workflow activities that are triggered when the Save button is clicked on the Financial Aid Cash Payment/Other form.

Example: Your institution requires multiple payments if the cash amounts is above $800. You create a workflow that validates the number of payments when the form is saved with a cash amount above $800.</td></tr>
</table>

SIS Saved Events - Entity Level

Saved events are triggered just after data has been saved to the database and are most often used to perform some additional activity such as creating a Contact Manager activity, triggering a document, or adding a student to a group. Saved events are only generated when one of the "trigger" fields is updated. The events are captured at the database trigger level.

Saved events are only visible in the Event Log of the Windows Service NextGen Nexus Event Workflows. Saved event workflows must be stored on a host that has a direct database connection such as the COM server. The workflow Add Students to a Group is an example of a workflow triggered by a Saved event.

Note: Saved events are triggered off a single main database table, therefore, entity mappings to items in other tables is not always available in the Saved event data.

Forms can be accessed from multiple paths and some fields exist in multiple forms. This table does not does not list all possible paths and field occurrences.

The following table lists the SIS Saved events at the **entity level**, sorted by Contract Entities.

SIS Saved Events - Entity Level

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Crm > Task		
Task		
Cmc.Nexus.Contracts > Cmc.Nexus.Crm > Task	Contact Manager > Activities (Add/Edit) (frmCmTask)	Cmc.Nexus.Crm • Task
Cmc.Nexus > Group Membership		
GroupMembership		
Cmc.Nexus.Contracts > Cmc.Nexus > Group Membership	View > Student Groups (frmSyStudentGroups)	Cmc.Nexus • GroupMembership
This event enables you to create an activity that is triggered when a student is added to or removed from a Student Group. Workflow example: Add Students to a Group.		
Cmc.Nexus > Person		
SyStudent Event		
Cmc.Nexus.Contracts > Cmc.Nexus > Person	Student > Student Master (frmAmStudMaster or frmAMStudMasterShort)	Cmc.Nexus • Person

Contracts	Anthology Student Form	Entity Mapping
This event enables you to create workflow activities that are triggered when a value in any field of the SyStudent table is changed. **Note**: Anthology Student databases and much of the business logic send updates to the SyStudent table multiple times due to triggers, related processes, etc. Therefore, multiple activities can be triggered by one change in the SyStudent table. To prevent this from happening, in your workflow make sure that a field actually changed before performing any activity on the event. Use the `HasChanged` method to ensure that the property you care about has actually been modified. See Checking for Record Inserts and Changes. Workflow example: Add Students to a Group, which is triggered when the veteran status is changed in the SyStudent table.		
Cmc.Nexus > Person Document		
PersonDocument		
Cmc.Nexus.Contracts > Cmc.Nexus > Person Document	Contact Manager > Documents (frmAmStudDocuments)	Cmc.Nexus • PersonDocument
Cmc.Nexus.Sis.Academics > Student Course		
StudentCourse		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Course	Student Bar: Academic Records > Schedule (frmAdEnrollSched) or Student Bar: Academic Records > Attendance (frmAdEnrollAttend)	Cmc.Nexus.Sis.Academics • StudentCourse
Cmc.Nexus.Sis.Academics > Student Enrollment Period		
StudentEnrollmentPeriod		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Enrollment Period	Academic Records > Enrollment	Cmc.Nexus.Sis.Academics • StudentEnrollmentPeriod
Cmc.Nexus.Sis.CareerServices > Student Employment History		
Student Employment History		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.CareerServices > Student Employment History	Career Services > Placement (frmPlPlacements)	Cmc.Nexus.Sis.CareerServices • New instance or update to StudentEmploymentHistory
This event enables you to create an activity that is triggered when a Student's employment history record is added or updated. Example: An employer is associated with the placement record.		
Cmc.Nexus.Sis.CareerServices > Student Placement Skill		

Contracts	Anthology Student Form	Entity Mapping
Student Placement Skill		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.CareerServices > Student Placement Skill	Career Services > Placement (frmPlPlacements)	Cmc.Nexus.Sis.CareerServices • New instance or update to StudentPlacementSummary
This event enables you to create an activity that is triggered when a student's placement skill record is added or updated. Example: A placement skill is added to a student record.		
Cmc.Nexus.Sis.FinancialAid > IsirMatch		
IsirMatch		
Cmc.Nexus.FinancialAid.Contracts > Cmc.Nexus.FinancialAid.Entities > Isir Matches (IsirMatchEntity)	Daily > Financial Aid > Import Data > Application Data (select Update Now) (Module: ISIRImport1) > Process Daily > Financial Aid > ISIR Matching (Module: ISIRMatchI2) > Auto Match or Manual Match View > Financial Aid > ISIR (Module: ISIRReceived)	Cmc.Nexus.FinancialAid.Services • IsirMatch
This event enables you to create an activity that is triggered when an ISIR is matched to a student record. The ISIR can be matched to the student by several processes in Anthology Student: • During ISIR import (Daily > Financial Aid > Import Data > Application Data (select Update Now) (Module: ISIRImport1) > Process). ISIRs are also processed in the back end at a later time by the Windows Service for Global ISIR processing and if Update Now not selected during ISIR Import. • Using the ISIR matching wizard (Daily > Financial Aid > ISIR Matching > Auto Match or Manual Match). • When the ISIR form is loaded (View > Financial Aid > ISIR). The IsirMatch event provides access to the fields from the IsirMatch entity.		
Cmc.Nexus.Sis.FinancialAid > Student Academic Year		
StudentAcademicYear		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Academic Year	Financial Aid > Packaging (frmFaStudAcadYears)	Cmc.Nexus.Sis.FinancialAid • StudentAcademicYear
Cmc.Nexus.Sis.FinancialAid > Student Grant Detail		
Fund Source - Grant		

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Grant Detail	Financial Aid > Packaging > Add New Source of Aid (frmFaStudGrant)	Cmc.Nexus.Sis.FinancialAid - New instance or update to StudentGrantDetail
This event enables you to create an activity that is triggered when a Grant Fund Source is added to a student's financial aid package or posted as a ledger transaction. **Note**: PaidDisbursements and ScheduledDisbursements collections cannot be mapped in the Saved event. This can only be done during the Saving event.		
Cmc.Nexus.Sis.FinancialAid > Student Loan Detail		
Fund Source - Loan		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Loan Detail	Financial Aid > Packaging > Add New Source of Aid (FaStudDirectLoan)	Cmc.Nexus.Sis.FinancialAid - New instance or update to StudentLoanDetail
This event enables you to create an activity that is triggered when a Loan Fund Source is added to a student's financial aid package or posted as a ledger transaction.		
Cmc.Nexus.Sis.FinancialAid > Student Award Pell		
Dependency Status		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Award Pell	Student > FAFSA or ISIR data can change the dependency status to change on multiple forms in Anthology Student	Cmc.Nexus.Sis.FinancialAid - StudentAwardSummary
This event enables you to create an activity that is triggered when a student's dependency status changes.		
Cmc.Nexus.Sis.StudentAccounts > Account Charge Transaction		
Student Ledger - Charge Transaction		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Account Charge Transaction	Student Accounts > Ledger Cards > Post Charges (frmSaTransTrxs)	Cmc.Nexus.Sis.StudentAccounts - New instance of AccountTransaction is created > AccountChargeTransaction
This event enables you to create an activity that is triggered when a Charge Transaction is posted to the student's ledger. The event message contains the amount, fund source, and date. You can use this information to build various workflow activities. Example: Send an SMS when new charges have been posted to a student's account.		
Cmc.Nexus.Sis.StudentAccounts > Account Payment Transaction		
Student Ledger - Payment Transaction		

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Account Payment Transaction	Student Accounts > Ledger Card > Post Payments (frmSaTransPayment)	Cmc.Nexus.Sis.StudentAccounts • New instance of AccountTransaction is created > AccountPaymentTransaction
This event enables you to create an activity that is triggered when a Payment Transaction is posted to the student's ledger. The event message contains the amount, fund source, and date. You can use this information to build various workflow activities. Examples: Send an email thanking the student for submitting a payment.		
Cmc.Nexus.Sis.StudentAccounts > Account Summary		
Account Summary		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Account Summary	Student Accounts > Ledger Card (frmSaLedger62)	Cmc.Nexus.Sis.StudentAccounts • AccountSummary
Cmc.Nexus.Sis.StudentAccounts > Collection Account		
Collection Account		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Collection Account	View > Student Accounts > Collections (frmSaCollections)	Cmc.Nexus.Sis.StudentAccounts • CollectionAccount
Cmc.Nexus.Sis.StudentAccounts > Student Payment Plan Statement		
Account Statement		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Student Payment Plan Statement	Daily > Student Accounts > Student Billing Statements > Student Payment Statements (frmSaPrintStatements)	Cmc.Nexus.Sis.StudentAccounts • StudentPaymentPlanStatement
This event enables you to create an activity that is triggered when a student's account statement is processed.		
Cmc.Nexus.Sis > Student Advisor		
Student Advisor		

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Contracts > Cmc.Nexus.Sis > Student Advisor	Add or edit advisor for an enrollment from the following forms: • View > Academic Records > Enrollment • View > Financial Aid > Packaging • View > Student Accounts > Ledger Card • View > Career Services > Placements • View > Loan Mangement > Loan Management • View > Contact Manager> Advisors • View > Contact Manager > International • Daily > Contact Manager > Advisor Assignment	Cmc.Nexus.Sis • StudentAdvisor
This event enables you to create workflow activities that are triggered when a value in the SyAdvisorByEnroll table is added or changed.		
Cmc.Nexus.Sis > Student Extra Curricular		
Extra-Curricular Activities		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis > Student Extra Curricular	Student > Student Master (frmAmStudMaster or frmAMStudMasterShort)	Cmc.Nexus.Sis • StudentExtraCurricular
This event enables you to create workflow activities that are triggered when a value in the Extra-Curricular activities field on the Student Master SyStudent.AmExtraCurrID is added or changed. Example: A workflow creates a Contact Manager activity for the Athletics Department when a student completes an application and chooses a sport that populates the Extra-Curricular field on the Student Master form. The Athletics Department then starts the interview process for sports teams.		

SIS Saved Events - Field Level

Saved events are triggered just after data has been saved to the database and are most often used to perform some additional activity such as creating a Contact Manager activity, triggering a document, or adding a student to a group. Saved events are only generated when one of the "trigger" fields is updated. The events are captured at the database trigger level.

Saved events are only visible in the Event Log of the Windows Service NextGen Nexus Event Workflows. Saved event workflows must be stored on a host that has a direct database connection such as the COM server. The workflow Add Students to a Group is an example of a workflow triggered by a Saved event.

Note: Saved events are triggered off a single main database table, therefore, entity mappings to items in other tables is not always available in the Saved event data.

The following table lists the SIS Saved events at the **field level**, sorted by Contract Entities.

SIS Saved Events - Field Level

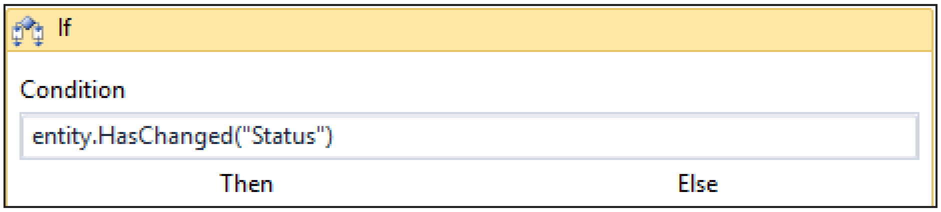
Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Crm > Task		
Task Status Id		
Cmc.Nexus.Contracts > Cmc.Nexus.Crm > Task	Contact Manager > Activities (Add/Edit) (frmCmTask)	Cmc.Nexus.Crm - Task.TaskStatusId
This event enables you to create a Contact Manager activity that is triggered when the Activity Status is changed (TaskStatusId field).		
Cmc.Nexus > Person		
Lead Type Id		
Cmc.Nexus.Contracts > Cmc.Nexus > Person	Student > Student Master (frmAmStudMaster or frmAMStudMasterShort)	Cmc.Nexus.Sis.Admissions - Prospect.LeadTypeId
This event enables you to create a Contact Manager activity that is triggered when the Applicant Type is changed (LeadTypeId field).		
Cmc.Nexus > Person Document		
Document Status		
Cmc.Nexus.Contracts > Cmc.Nexus > Person Document	Contact Manager > Documents (frmAmStudDocuments)	Cmc.Nexus - PersonDocument.DocumentStatusId

<table>
<tr><th>Contracts</th><th>Anthology Student Form</th><th>Entity Mapping</th></tr>
<tr><td colspan="3">This event enables you to create an activity that is triggered when the Document Status is changed on a single document or list of documents. You can use the LookupReferenceItem activity to identify changed documents.
Example: When an application is received, a workflow activity causes an email to be sent to the Dean of Admissions.</td></tr>
<tr><td colspan="3"><b>Cmc.Nexus.Sis.Academics > Student Course</b></td></tr>
<tr><td colspan="3">Course Status</td></tr>
<tr><td>Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Course</td><td>Student Bar: Academic Records > Schedule (frmAdEnrollSched) or Student Bar: Academic Records > Attendance (frmAdEnrollAttend)</td><td>Cmc.Nexus.Sis.Academics
- StudentCourse.Status</td></tr>
<tr><td colspan="3">
This event enables you to create activities that are triggered when a student's Course Status is changed.

Course Status values in Anthology Student are Future, Scheduled, Current, Dropped, Completed, Reserved, Waitlisted, and Leave of Absence. An event is raised each time a Course Status value changes.

Note: During certain Course Status changes multiple events may be triggered. To avoid duplication of workflow activities, add the following condition at the top of your workflow:

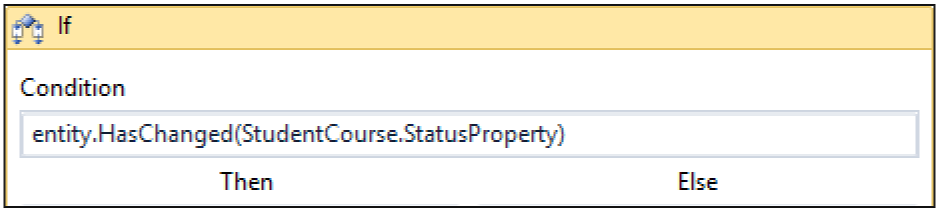
```
entity.HasChanged("Status")
```



entity.HasChanged("Status")

Another, more accurate, option is to specify the property of the entity that has changed:

```
entity.HasChanged(StudentCourse.StatusProperty)
```



entity.HasChanged(StudentCourse.StatusProperty)

You may also want to validate the current Status you are looking for with an additional condition.

For more information, see Check for Record Inserts and Changes.
</td></tr>
<tr><td colspan="3">Grade Status and Grade Letter</td></tr>
</table>

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Course	Academic Records > Final Grades (frmAdEnrollGrades)	Cmc.Nexus.Sis.Academics • StudentCourse.Status • StudentCourse.LetterGrade
This event enables you to create an activity that is triggered when Grade Status or Grade Letter is changed on a student's course record. **Note**: The event is raised by any letter grade changes, not just changes from "I" (incomplete) to "F" (fail).		
Final Grade		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Course	Academic Records > Final Grades (frmAdEnrollGrades)	Cmc.Nexus.Sis.Academics • StudentCourse.LetterGrade • StudentCourse.NumericGrade • StudentCourse.UnitValues • StudentCourse.GradePoints
This event is raised when a Course Grade is posted on the Final Grade form. The event is raised by updates in the AdEnrollSched table. Example: When a letter grade changes from "B" to "A", a congratulatory note is sent to the student. Workflow example: Check if a Grade was Posted.		
Cmc.Nexus.Sis.Academics > Student Enrollment Period		
Enrollment Status		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Enrollment Period	Academic Records > Enrollment > Date/Status tab (frmAdEnroll)	Cmc.Nexus.Sis.Academics • StudentEnrollmentPeriod.EnrollmentStatusId
This event enables you to create an activity that is triggered when the Enrollment Status is changed on a student's record. Workflow examples: Charging a Fee when Enrollment Status Changes and Register Students into a Course		
Grade Level		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Enrollment Period	Academic Records > Enrollment > Progress tab (frmAdEnroll)	Cmc.Nexus.Sis.Academics • StudentEnrollmentPeriod.GradeLevelId
This event enables you to create an activity that is triggered when the Grade Level is changed on a student's record.		
Graduation Date		

Contracts	Anthology Student Form	Entity Mapping
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.Academics > Student Enrollment Period	Academic Records > Enrollment - Date/Status tab (frmAdEnroll)	Cmc.Nexus.Sis.Academics • StudentEnrollmentPeriod.GraduationDate
This event is raised when a Graduation Date changes. You can use this event to trigger an activity to inform students of a change in their eligibility for graduation.		
Cmc.Nexus.Sis.FinancialAid > Student Academic Year		
Packaging Status		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.FinancialAid > Student Academic Year	Financial Aid > Packaging (frmFaStudAcadYears)	Cmc.Nexus.Sis.FinancialAid • StudentAwardSummary
This event enables you to create an activity that is triggered when the Student's financial aid Packaging Status changes on a student's record. Examples: • A workflow is triggered when a financial aid Packaging Status changes from Partial Packaged to Final Package. • A workflow is triggered when a financial aid Packaging Status changes from Not Packaged to Cash. **Note**: Changes to Budget Items will trigger a separate event from changes to main Student Academic Year event entity. For example, if you change Housing from Off Campus to On Campus, you will receive two events - one for housing status change on the main entity and one for changes to the Budget Items collection for Room and Board.		
Cmc.Nexus.Sis.StudentAccounts > Account Summary		
Account Status		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Account Summary	Student Accounts > Ledger Card (frmSaLedger62)	Cmc.Nexus.Sis.StudentAccounts • AccountSummary. AccountStatusDetail. AccountStatusId
This event enables you to create an activity that is triggered when the Account Status is changed on a student's ledger card.		
Cmc.Nexus.Sis.StudentAccounts > Collection Account		
Collection Status		
Cmc.Nexus.Contracts > Cmc.Nexus.Sis.StudentAccounts > Collection Account	View > Student Accounts > Collections (frmSaCollections)	Cmc.Nexus.Sis.StudentAccounts • CollectionAccount. AccountStatusDetail. AccountStatusId

Contracts	Anthology Student Form	Entity Mapping
This event enables you to create an activity that is triggered when a Collection Status value is changed on a student's enrollment.		

Time-based Events

Time-based events are recurring events that aree triggered based on predefined intervals. These events are usually triggered based on Windows services.

To trigger time-based events in your workflow, include the Cmc.Domain.Entities.Sis.SisSchedule entity when you create or define the workflow. The SisSchedule entity includes three events that are triggered in specific time intervals.

Time-based Events

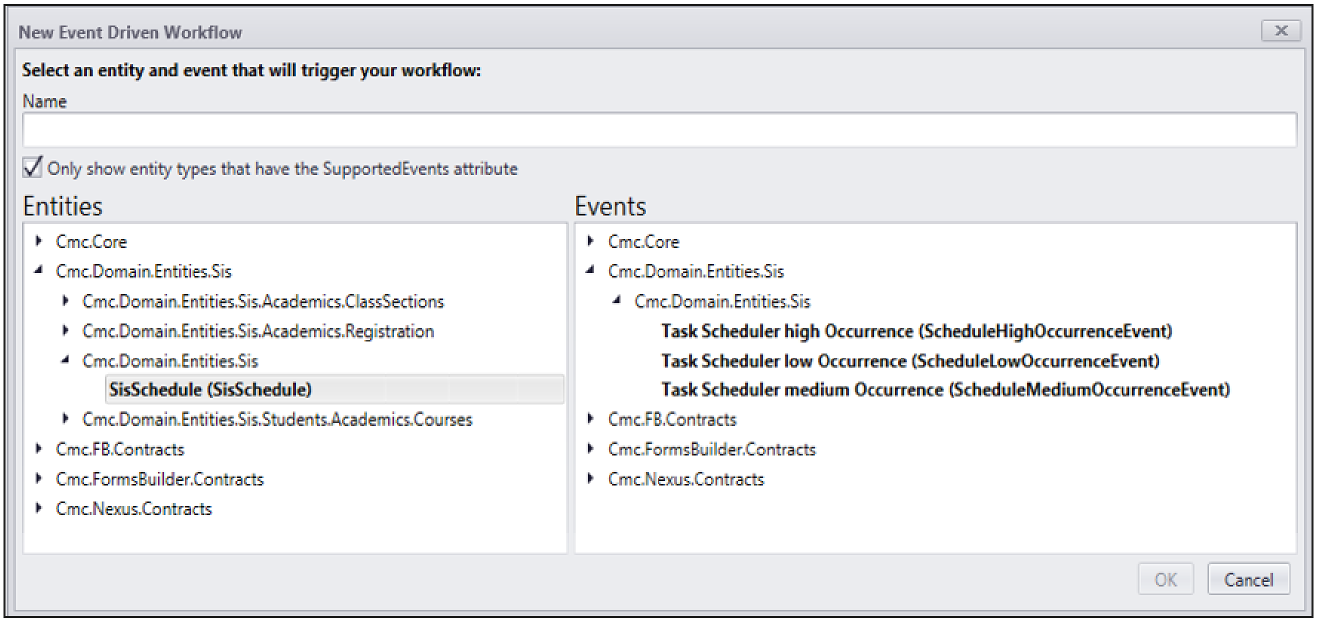
Entity	Event	Occurrence
SisSchedule	ScheduleHighOccurenceEvent	Every 6 seconds
SisSchedule	ScheduleMediumOccurenceEvent	Every 6 hours
SisSchedule	ScheduleLowOccurenceEvent	Every 24 hours

In a time-based workflow, you do not need to have any entity-related information as in other workflows. You need to include the following activities to get context information:

- ExecuteDataReader
- ExecuteNonQuery
- ExecuteQuery

To create a time-based event workflow:

1. Start Workflow Composer.
2. Click **New Event Workflow**. The New Event Driven Workflow window is displayed.
3. Specify a **Name** for the workflow.
4. In the Entities pane, select the **SisSchedule** entity, and select the appropriate event in the Events pane.



Select an entity and event that will trigger your workflow:

Name

☑ Only show entity types that have the SupportedEvents attribute

Entities

- ▸ Cmc.Core
- ◢ Cmc.Domain.Entities.Sis
  - ▸ Cmc.Domain.Entities.Sis.Academics.ClassSections
  - ▸ Cmc.Domain.Entities.Sis.Academics.Registration
  - ◢ Cmc.Domain.Entities.Sis
    - **SisSchedule (SisSchedule)**
  - ▸ Cmc.Domain.Entities.Sis.Students.Academics.Courses
- ▸ Cmc.FB.Contracts
- ▸ Cmc.FormsBuilder.Contracts
- ▸ Cmc.Nexus.Contracts

Events

- ▸ Cmc.Core
- ◢ Cmc.Domain.Entities.Sis
  - ◢ Cmc.Domain.Entities.Sis
    - **Task Scheduler high Occurrence (ScheduleHighOccurrenceEvent)**
    - **Task Scheduler low Occurrence (ScheduleLowOccurrenceEvent)**
    - **Task Scheduler medium Occurrence (ScheduleMediumOccurrenceEvent)**
- ▸ Cmc.FB.Contracts
- ▸ Cmc.FormsBuilder.Contracts
- ▸ Cmc.Nexus.Contracts

5. Click **OK**.

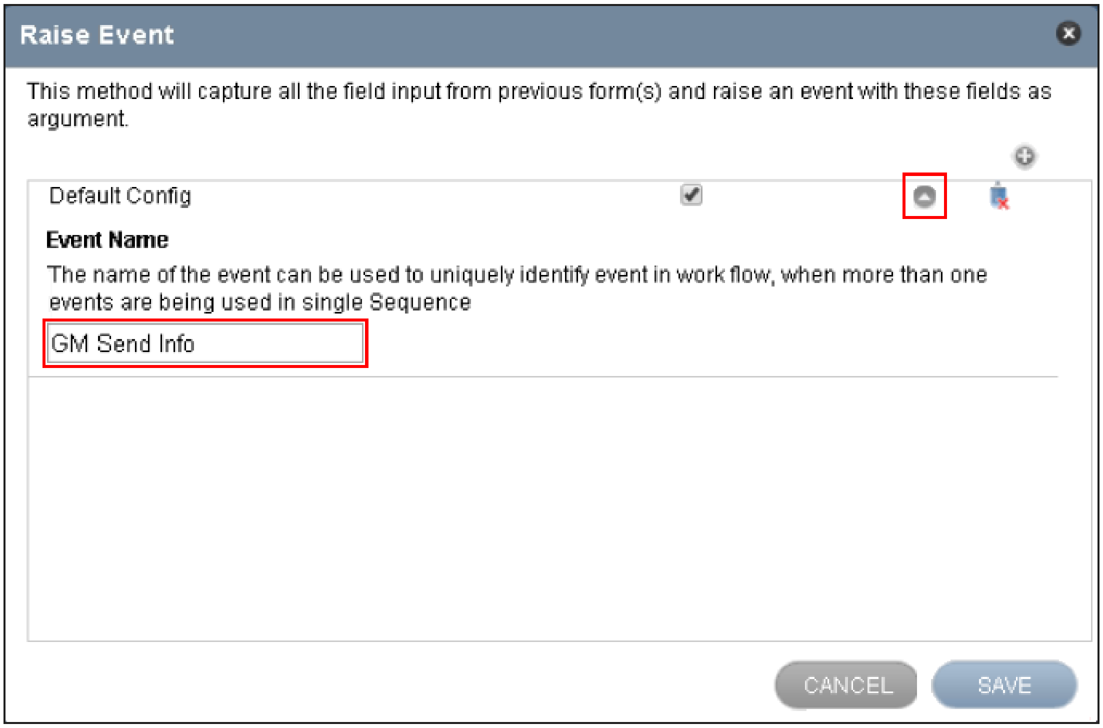
Forms Builder Events

This content is applicable to Forms Builder version 2.x only.

Forms Builder's eventing integration with Workflow enables you to raise events from Forms Builder sequences and capture these events using Workflow or any service bus. The information from the events can be used for validation, setting defaults, creating tasks for staff, and countless other purposes.

Raise Event Rule

Designers of sequences in Forms Builder can choose to raise an event between any form transition. Whenever the **Raise Event** rule is encountered, Forms Builder collects the field input from previous forms and raises an event with these fields as arguments.



Raise Event

This method will capture all the field input from previous form(s) and raise an event with these fields as argument.

Default Config ☑

Event Name

The name of the event can be used to uniquely identify event in work flow, when more than one events are being used in single Sequence

GM Send Info

The Raise Event rule has one optional configuration field for `EventName`. This field can be used to distinguish events coming from multiple Raise Event rules from the same or different sequences.

Event Details

Every event raised from Forms Builderr has some basic properties to work with. You can get details regarding the event and perform different actions in your workflow.

Forms Builder Events

Event	Property	Description

`FormEntity()`	`EventName():string`
	`Fields():IDictionary<string,string>`
	`UserId():int`
		The `FormEntity`:
		• holds the data coming in from Forms Builder. This data is arranged in key-value collections representing field names from Forms Builder forms and their corresponding values in text.
		• holds the optional `EventName` specified in the Raise Event rule configuration.
		Note: To capture a Forms Builder event in a workflow, specify the exact Event Name of the Raise Event rule. For example, in the condition field of an If activity, specify the following:
		`entity.EventName = “GM Send Info”`
		where "`GM Send Info`" is the Event Name in Forms Builder.
		• has a `UserID`, which is the unique identifier provided by STS for the logged in user.

Event	Property	Description

`FormTransitionEventArgs()`	`DefaultFields():IDictionary<string,string>`
	`ValidationMessages():ValidationMessageCollection`
		Subscribers to events from Forms Builder can communicate back to the Forms Builder sequence originating the event via the `FormTransitionEventArgs`.
		- `DefaultFields` can be used to set new defaults on upcoming forms or change values on previous forms. This property represents a collection of key-value pairs.
		- `ValidationMessages` can be used to return messages in response to forms validation.

Application Key IDs Used with Anthology Student

`FormEntity` contains different identities alongside all the Forms Builder fields being collected. These Ids are created while executing different Forms Builder rules. Some of the Ids are populated based upon the user type. For example, a student always has SyStudentId populated in StudentIdAppKey.

Application Key IDs

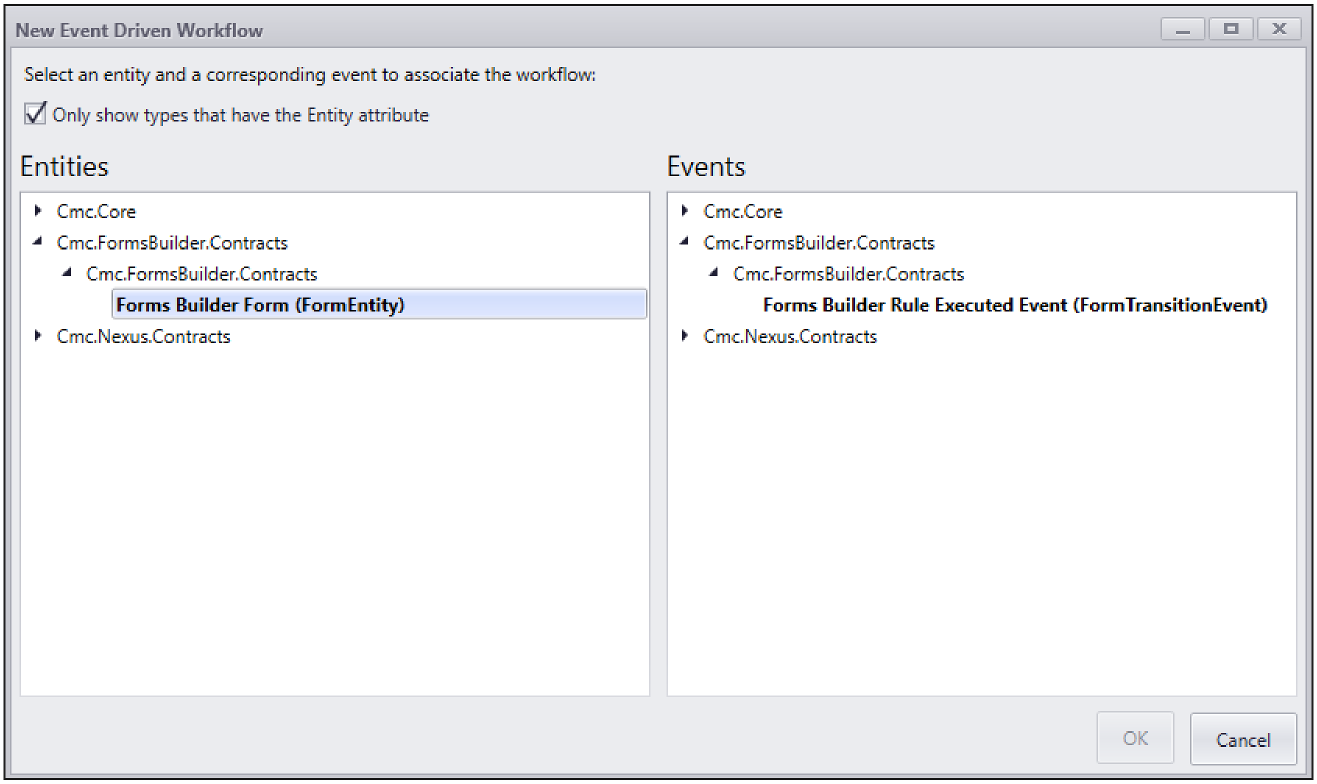
Field Name	Description
ApplicantEmailIdAppKey	Email Id
CampusIdAppKey	Campus Id
EnrollIdAppKey	Student Enrollment Id generated when new Enrollment is created
IsPaymentMadeAppKey	Is Payment Made
NumDuplicatesAppKey	Number of Duplicates generated from Duplicate Check Rule
OnlineApplicantIdAppKey	Online Applicant Id
PaymentAmpountAppKey	Payment Amount App Key
PaymentReceiptAppKey	Payment Receipt generated upon successful payment
PendingApplicantsAppKey	Pending Applicants Flag
PortalUserIdAppKey	Portal User Id points to wpUserID
StudentIdAppKey	Student Id points to SyStudentID
SyAddressIdAppKey	Address Id generated while saving profile information

Workflow for Forms Builder Events

Once the sequence in Forms Builder has been setup with the Raise Event rule, the next step is to create an event subscriber using Workflow

1. In Workflow, click on **New Event Workflow**.
2. Under Entities, expand **Cmc.FormsBuilder.Contracts** and select **Forms Builder Form (FormEntity)**.

   Under Events, expand **Cmc.FormsBuilder.Contracts** and **Forms Builder Rule Executed Event (FormTransitionEvent**.

   New Event Driven Workflow

   Select an entity and a corresponding event to associate the workflow:

   ☑ Only show types that have the Entity attribute

   Entities

   - Cmc.Core
   - Cmc.FormsBuilder.Contracts
     - Cmc.FormsBuilder.Contracts
       - **Forms Builder Form (FormEntity)**
   - Cmc.Nexus.Contracts

   Events

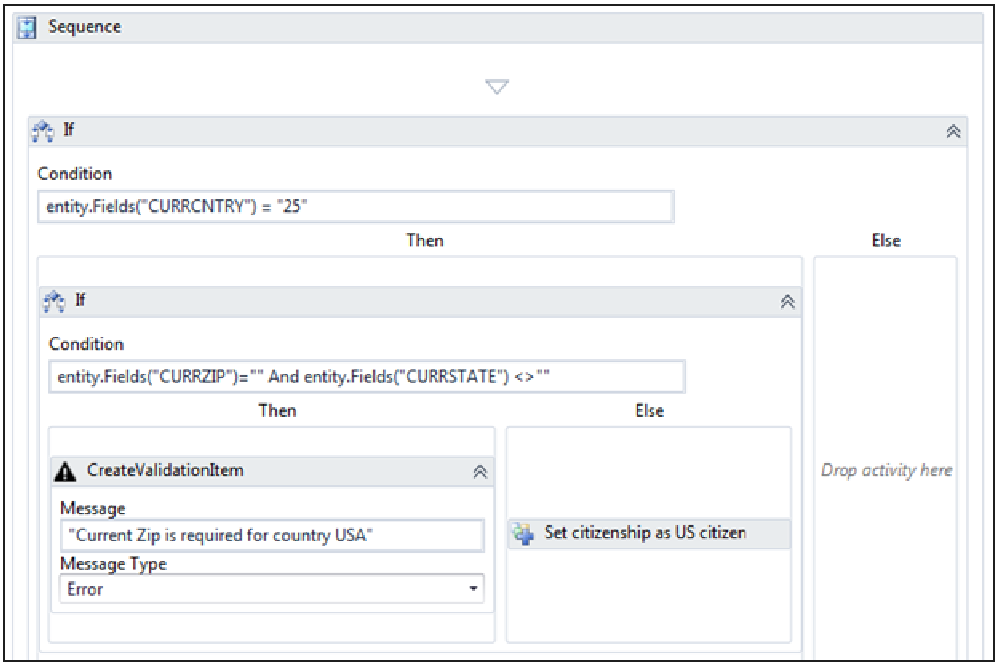
   - Cmc.Core
   - Cmc.FormsBuilder.Contracts
     - Cmc.FormsBuilder.Contracts
       - **Forms Builder Rule Executed Event (FormTransitionEvent)**
   - Cmc.Nexus.Contracts

3. Create your workflow using the Activities available in the Workflow Designer Toolbox.

   Example

   This workflow makes the ZIP code required when the country is USA and a State is selected. Otherwise, a validation message is created for the applicant.

The AddToCollection activity sets the default citizenship to "US citizen".



Condition: entity.Fields("CURRCNTRY") = "25"

Condition: entity.Fields("CURRZIP")="" And entity.Fields("CURRSTATE") <> ""

Message: "Current Zip is required for country USA"

Message Type: Error

Set citizenship as US citizen

For information on how to create event handlers for Forms Builder events, see Create Event Handlers in .NET.

To see how a Forms Builder event can be used in a workflow, see Populate Fields in a Forms Builder Form.

Create Event Handlers in .NET

This topic describes how to create a few simple event handlers for the `Person` entity to perform validations during the save process.

Subscribe to an Event

In this example we are working within the `Logic` project. This project contains our event handles and references the CMC framework and contracts that define the events.

Step 1: Add Required References

To utilize the CMC framework, you need to add a reference to `Cmc.Core.dll`.

1. Open the `EventHandlers.sln` solution in Visual Studio.
2. In Solution Explorer, right-click on **Logic\References** and select **Add Reference...**
3. On the Browse tab, click the **Browse...** button.
4. From the Select the files to reference... dialog, select `SDKPath\Cmc.Core.dll` and `SDKPath\Cmc.FormsBuilder.Contracts.dll` and click **Add**.
5. From the Reference Manager dialog, click **OK**.

Step 2: Make your Assembly Visible to the CMC Framework

To make types defined within this assembly discoverable by the CMC framework, we need to add the ExtensionAssembly assembly level attribute.

1. Within the Solution Explorer, open `Logic\Properties\AssemblyInfo.cs`.
2. Add the **[assembly: ExtensionAssembly]** attribute to the file.

```
//...
[assembly: ExtensionAssembly]
//...
```

Step 3: Create the EventSubscriber Type

During initialization, the EventService uses a container to discover all types that implement the IEventSubscriber interface. After discovery, the EventService invokes the RegisterHandlers method on each implementation of the interface, giving the implementer an opportunity to register event handlers.

The EventSubscriber type is an abstract class that simplifies the implementation of IEventSubscriber.

1. In Solution Explorer, right-click on the Logic project and select **Add -> Class...**
2. In the Name text box, enter **FormTransitionEventSubscriber.cs**.
3. Click the **Add** button.

4. Change the scope modifier of the newly added class to **internal**.
5. Inherit the class from **EventSubscriber**.
6. Click on the class name and pull down the smart tag to implement the abstract method, **RegisterHandlers**.

```
using Cmc.Core.Eventing;
namespace Logic
{
  internal class FormTransitionEventSubscriber : EventSubscriber
  {
    public override void RegisterHandlers(IEventService eventService)
    {
      throw new System.NotImplementedException();
    }
  }
}
```

Step 4: Register an Event Handler

Next, implement the RegisterHandlers method to register a handler for the FormTransitionEvent that validates a FormEntity instance prior to it being saved to the database.

1. Implement the abstract method RegisterHandlers to retrieve the SavingEvent from the provided IEventService. Register a handler for the Person type that does the following:
   a. Adds a validation message if there are no items in the Phones collection.
   b. Adds a validation message if there are no items in the Addresses collection.

```
using System;
using Cmc.Core.Eventing;
using Cmc.FormsBuilder.Contracts;

namespace Logic
{
  public class FormTransitionEventSubscriber : EventSubscriber
  {
    public override void RegisterHandlers(IEventService eventService)
    {
      eventService.GetEvent<FormTransitionEvent>().RegisterHandler<FormEntity>
((e, a) =>
      {
        if (e.Fields.ContainsKey("CURRCNTRY") && (e.Fields["CURRCNTRY"] == "25"))
        {
          if (String.IsNullOrEmpty(e.Fields["CURRZIP"]) && !String.IsNullOrEmpty
(e.Fields["CURRSTATE"]))
          {
            a.ValidationMessages.Add(new ValidationMessage("Current Zip is
required for country USA"));
```

```
          }
          else
          {
            a.DefaultFields["CITIZEN"] = "8"; // Default to US
          }
          }
          else
          {
            a.DefaultFields["CITIZEN"] = "3"; // Default to Non-US Citizen
          }
      });
    }
  }
}
```

Test the Library

Copy the `Logic\bin\Debug\Logic.dll` to the `bin` folder of your host application and create a Person without any phone numbers or addresses. When you save the Person, you should receive two errors.

Event Scheduling

Event scheduling enables you to schedule an event to occur based on a recurrence pattern. Event scheduling utilizes the Job Scheduler in SQL Server and the existing stored procedure `sproc_Notification_Timer_ScheduledEvent`.

The stored procedure takes the following arguments:

```
Entity: Schedule

Event: Schedule Occurrence Event

sproc_Notification_Timer_ScheduledEvent @key = ‘Birthdays’
```

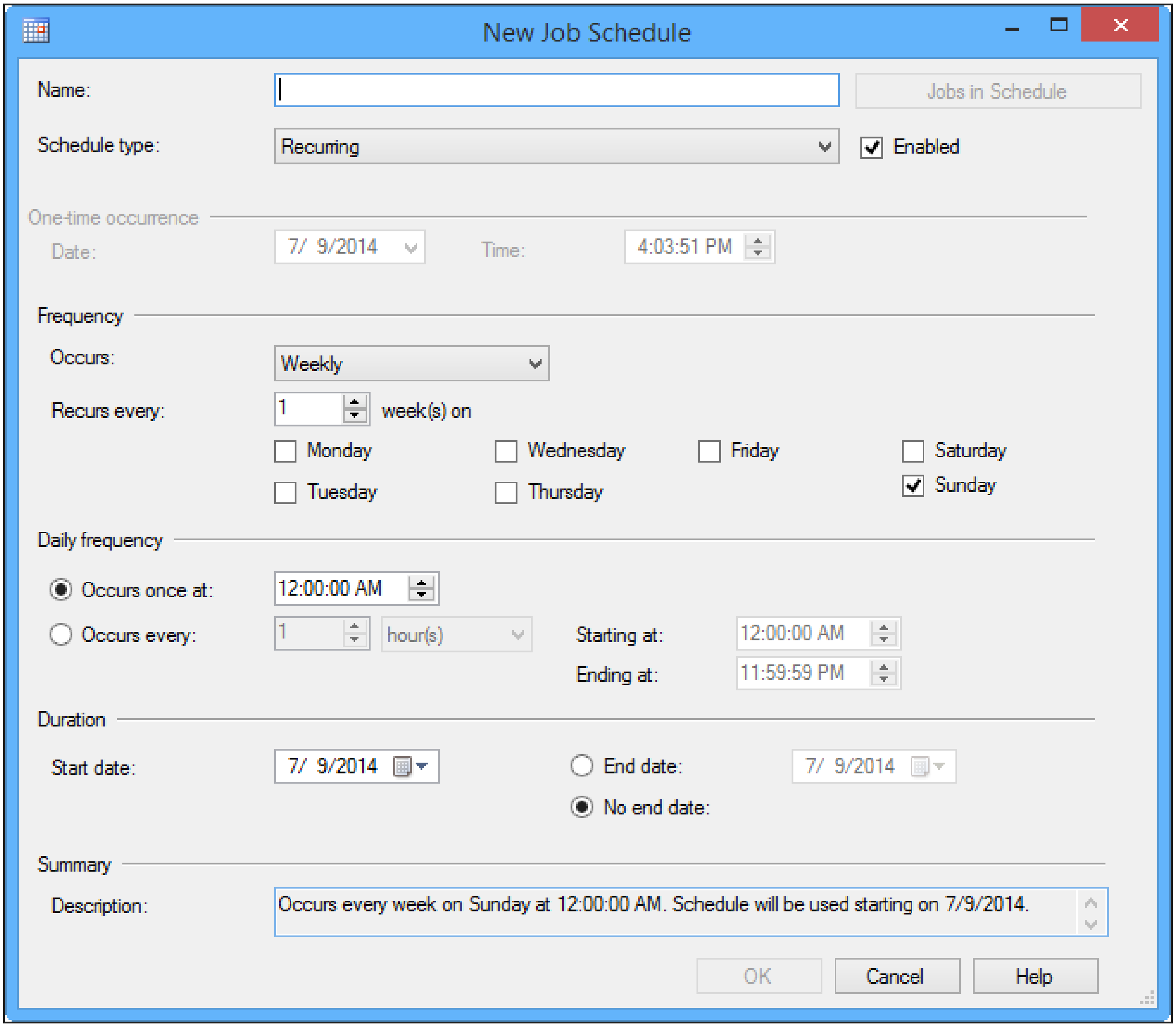
(example)

The stored procedure creates the job that can be scheduled in SQL Server Management Studio.

Create and Attach a Schedule to a Job in SQL Management Studio

1. In **Object Explorer**, connect to an instance of the SQL Server Database Engine, and then expand that instance.
2. Expand **SQL Server Agent**, expand **Jobs**, right-click the job you want to schedule, and click **Properties**.
3. Select the **Schedules** page, and then click **New**.
4. In the **Name** box, type a name for the new schedule.
5. Clear the **Enabled** check box if you do not want the schedule to take effect immediately following its creation.
6. For **Schedule Type**, click **Recurring**. Complete the Frequency, Daily Frequency, and Duration groups in

the New Job Schedule window.



New Job Schedule

Name:

Schedule type: Recurring

☑ Enabled

One-time occurrence

Date: 7/ 9/2014

Time: 4:03:51 PM

Frequency

Occurs: Weekly

Recurs every: 1 week(s) on

□ Monday □ Wednesday □ Friday □ Saturday

□ Tuesday □ Thursday ☑ Sunday

Daily frequency

● Occurs once at: 12:00:00 AM

○ Occurs every: 1 hour(s)

Starting at: 12:00:00 AM

Ending at: 11:59:59 PM

Duration

Start date: 7/ 9/2014

○ End date: 7/ 9/2014

● No end date:

Summary

Description: Occurs every week on Sunday at 12:00:00 AM. Schedule will be used starting on 7/9/2014.

Attach a Schedule to a Job

1. In **Object Explorer**, connect to an instance of the SQL Server Database Engine, and then expand that instance.

2. Expand **SQL Server Agent**, expand **Jobs**, right-click the job that you want to schedule, that is, the job created by the stored procedure, and click **Properties**.

3. Select the **Schedules** page, and then click **Pick**.

4. Select the schedule you want to attach and click **OK**.

5. In the **Job Properties** dialog box, double-click the attached schedule.

6. Verify that **Start date** is set correctly. If it is not, set the date when you want for the schedule to start, and then click **OK**.
7. In the **Job Properties** dialog box, click **OK**.

Sample Workflows

The sample workflow presented in this section were developed prior to the introduction of the new Anthology object model. To adapt these workflows to the new object model, please refer to Legacy Workflows.

Add Students to a Group

In this example, an institution wants to manage its military students by groups. Whenever a student's veteran status is set to "Yes", the student is added to a group called "Military Students". If a student's veteran status changes to "No", the workflow removes the student from that group.

1. In the standard interface of Anthology Student, create a student group as follows:

    a. Select the **Groups** tile. The list of existing Student Groups is displayed.

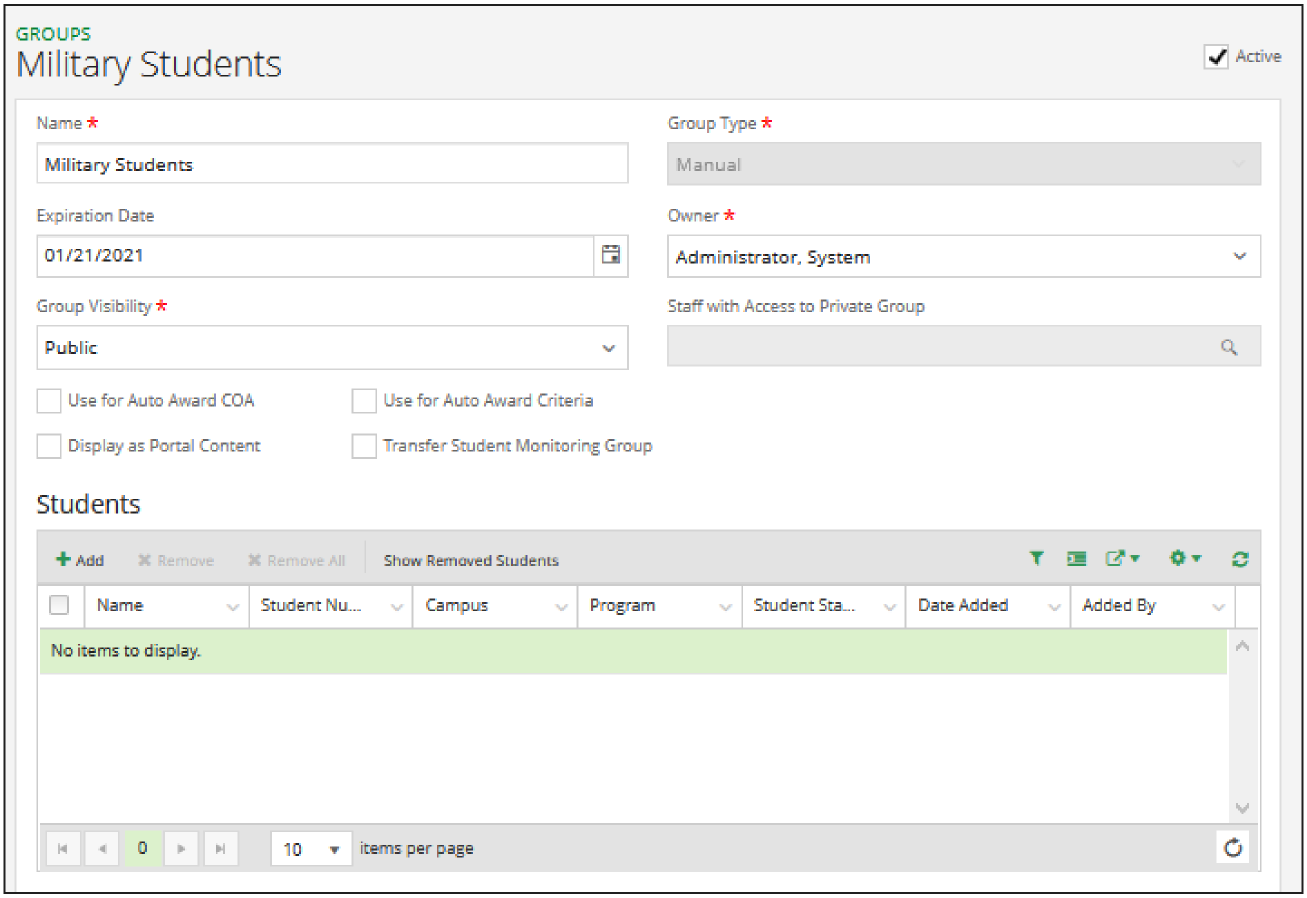
    b. Click **New**.

    c. In the Group Name text box, specify a name, e.g., **Military Students**.

    d. Select an appropriate **Expiration Date**. Keep in mind that this will be a long running workflow.

    e. Select values for additional group properties or accept the defaults.

    f. Click **Save & Close**.



GROUPS

Military Students

☑ Active

Name *: Military Students

Group Type *: Manual

Expiration Date: 01/21/2021

Owner *: Administrator, System

Group Visibility *: Public

Staff with Access to Private Group:

□ Use for Auto Award COA
□ Use for Auto Award Criteria
□ Display as Portal Content
□ Transfer Student Monitoring Group

Students

□	Name	Student Nu...	Campus	Program	Student Sta...	Date Added	Added By
No items to display.							

0 · 10 items per page

In the legacy interface of Anthology Student, create a student group as follows:

a. Navigate to **View > Student Groups**. The Student Groups form is displayed.

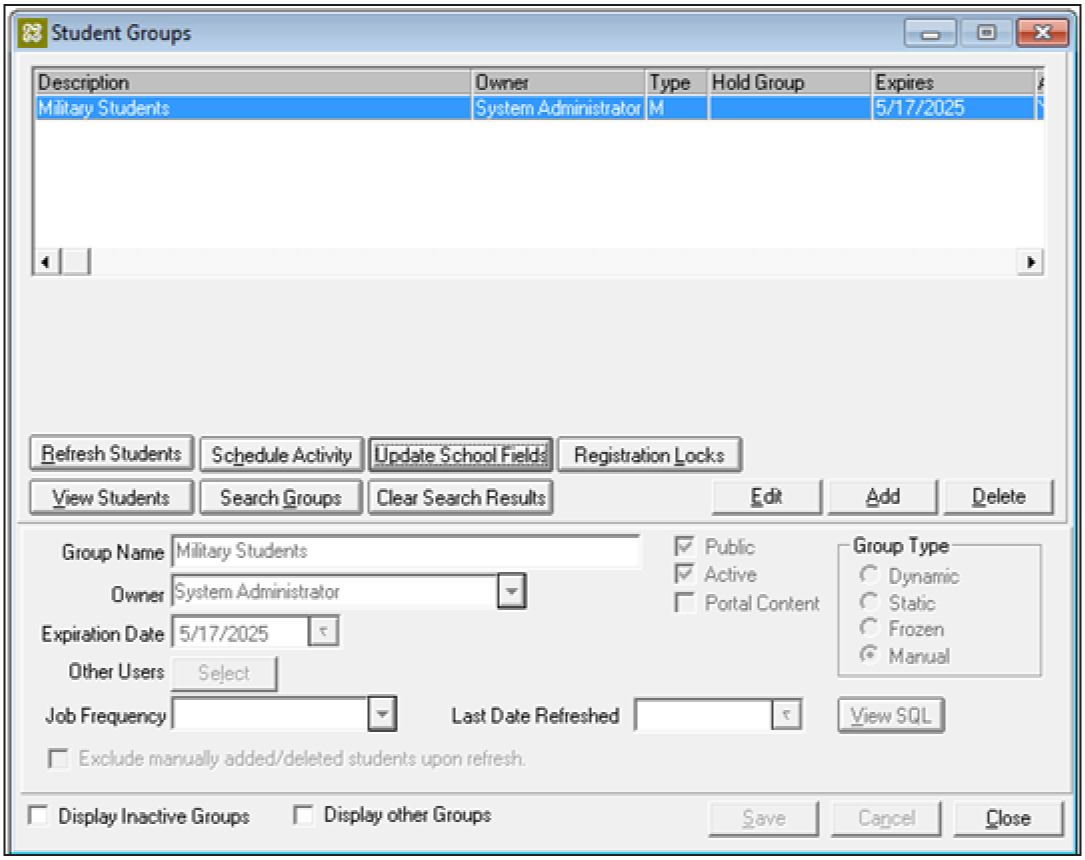
b. Click **Add**.

c. In the Group Name text box, specify a name, e.g., **Military Students**.

d. Select an appropriate **Expiration Date**. Keep in mind that this will be a long running workflow.

e. Select values for additional group properties or accept the defaults.

f. Click **Save** and **Close**.



Description	Owner	Type	Hold Group	Expires
Military Students	System Administrator	M		5/17/2025

Group Name: Military Students

Owner: System Administrator

Expiration Date: 5/17/2025

Other Users

Job Frequency:

Last Date Refreshed:

☑ Public

☑ Active

□ Portal Content

Group Type

○ Dynamic

○ Static

○ Frozen

● Manual

□ Exclude manually added/deleted students upon refresh.

□ Display Inactive Groups □ Display other Groups

2. Start the **Workflow** application from your desktop.

3. Click **New Event Workflow**.

4. In the Entities area:

   a. Click ▶ next to **Cmc.Nexus.Contracts**.

   b. Click ▶ next to **Cmc.Nexus**.

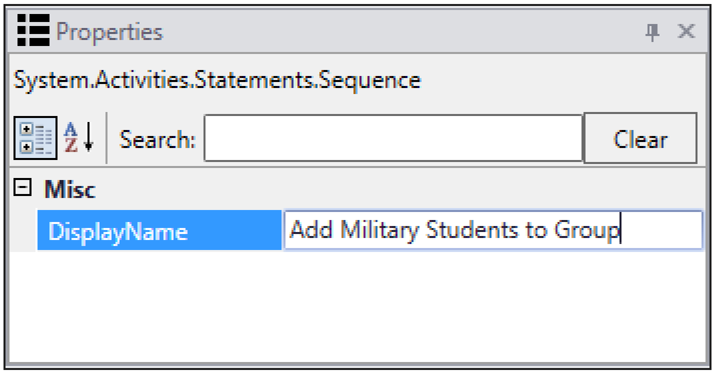
   c. Click **Person (Person)**.

5. In the Events area, click **Saved (SavedEvent)**.

6. Specify a **Name** for the workflow and click **OK**.

7. In the Properties pane of the Designer, specify **Add Military Students to Group** as the DisplayName of the sequence.

   Note: It is a good practice to assign a meaningful DisplayName to each activity as soon as it is dragged into the workflow. The DisplayName makes it easier to track the workflow in log files and reports.

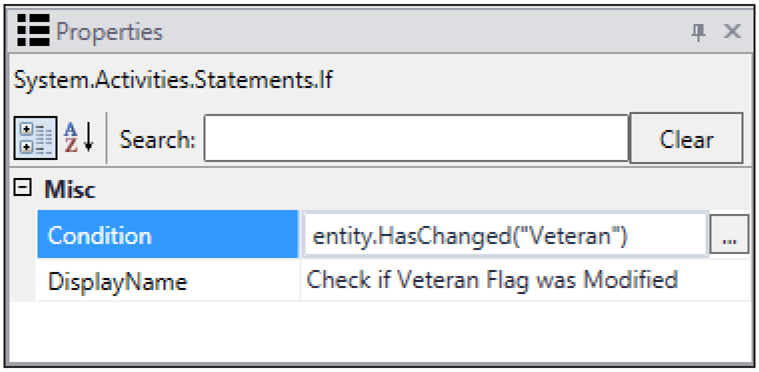
   

   System.Activities.Statements.Sequence

   DisplayName: Add Military Students to Group

8. In the Toolbox, under Control Flow, select the **If** activity and drag it into the sequence.
9. In Properties pane, specify **Check if the Veteran Flag was Modified** as the DisplayName of the If condition.
10. In the Condition field, specify the following VB expression: **entity.HasChanged("Veteran")**

    Refer to Helpful Hints to learn more about the purpose of this condition.

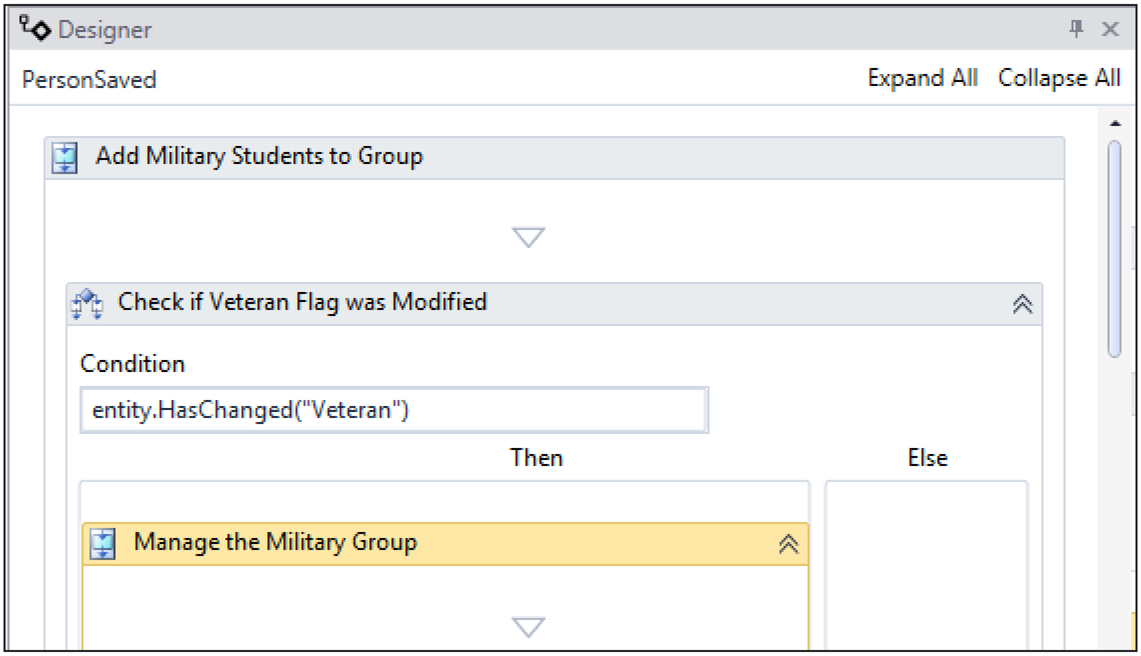
    

    System.Activities.Statements.If

    Condition: entity.HasChanged("Veteran")

    DisplayName: Check if Veteran Flag was Modified

11. In Properties pane, specify **Manage the Military Group** as the DisplayName of the Then sequence in the If condition.


Designer
PersonSaved
Expand All
Collapse All
Add Military Students to Group
Check if Veteran Flag was Modified
Then
Else
Manage the Military Group

Condition: entity.HasChanged("Veteran")

12. In the Toolbox, under Cmc.Nexus.Workflow, select the **LookupStudentGroup** activity and drag it into the "Then" branch of the "If" condition.

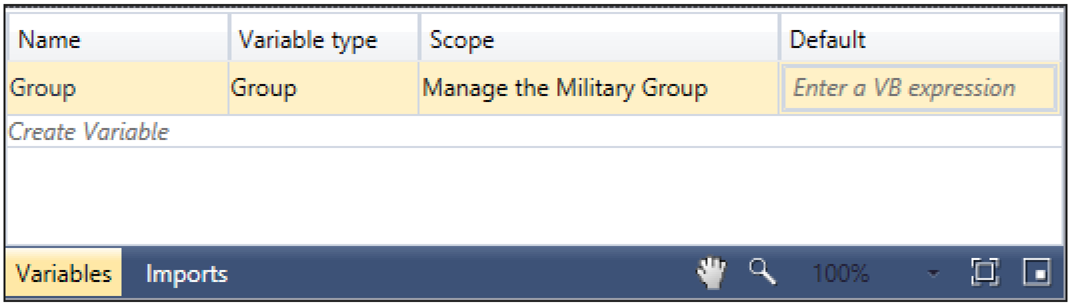
13. In Properties pane, specify **Find Military Students Group** as the DisplayName of the LookupStudentGroup activity.

14. Create a variable to pass the GroupId to the activity that will add or remove students from the group.

    a. Click the **Variables** tab in the Designer pane.

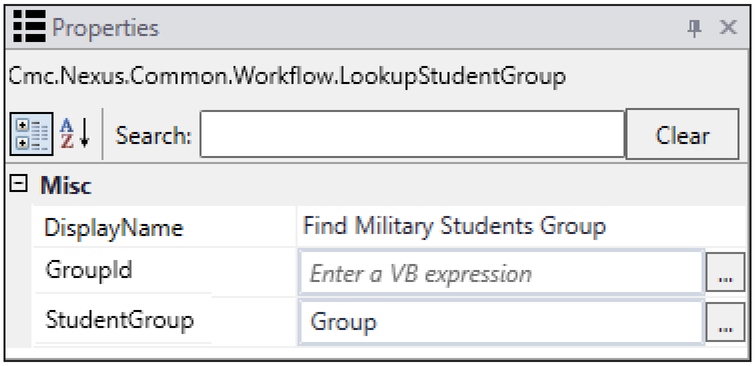
    b. Add the variable name **Group**.

    c. Choose the **Variable type**. For groups, it is found under Cmc.Nexus.Group.



Name	Variable type	Scope	Default
Group	Group	Manage the Military Group	*Enter a VB expression*
Create Variable			

    d. In the Properties pane of the LookupStudentGroup activity, specify **Group** as the Name of the variable in the StudentGroup field.



Cmc.Nexus.Common.Workflow.LookupStudentGroup

Misc

DisplayName	Find Military Students Group
GroupId	*Enter a VB expression*
StudentGroup	Group

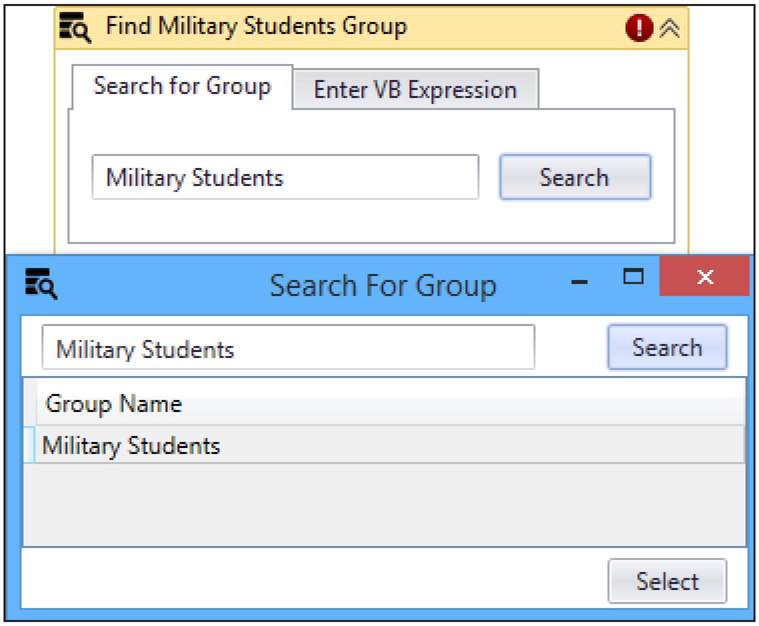
15. Use the LookupStudentGroup activity to search your Anthology Student system for groups and select the group created in step 1.

    a. Specify **Military Students** in the Search for Group tab of the LookupStudentGroup activity.

    b. Click **Search**.

    c. In the Search for Group window, select the **Military Students** group from the returned list of groups.

    d. In the Search for Group window, click **Select**.



Military Students

Military Students

Group Name
Military Students

16. In the Toolbox, under Control Flow, select another **If** activity and drag it into the Then sequence of the first If condition.

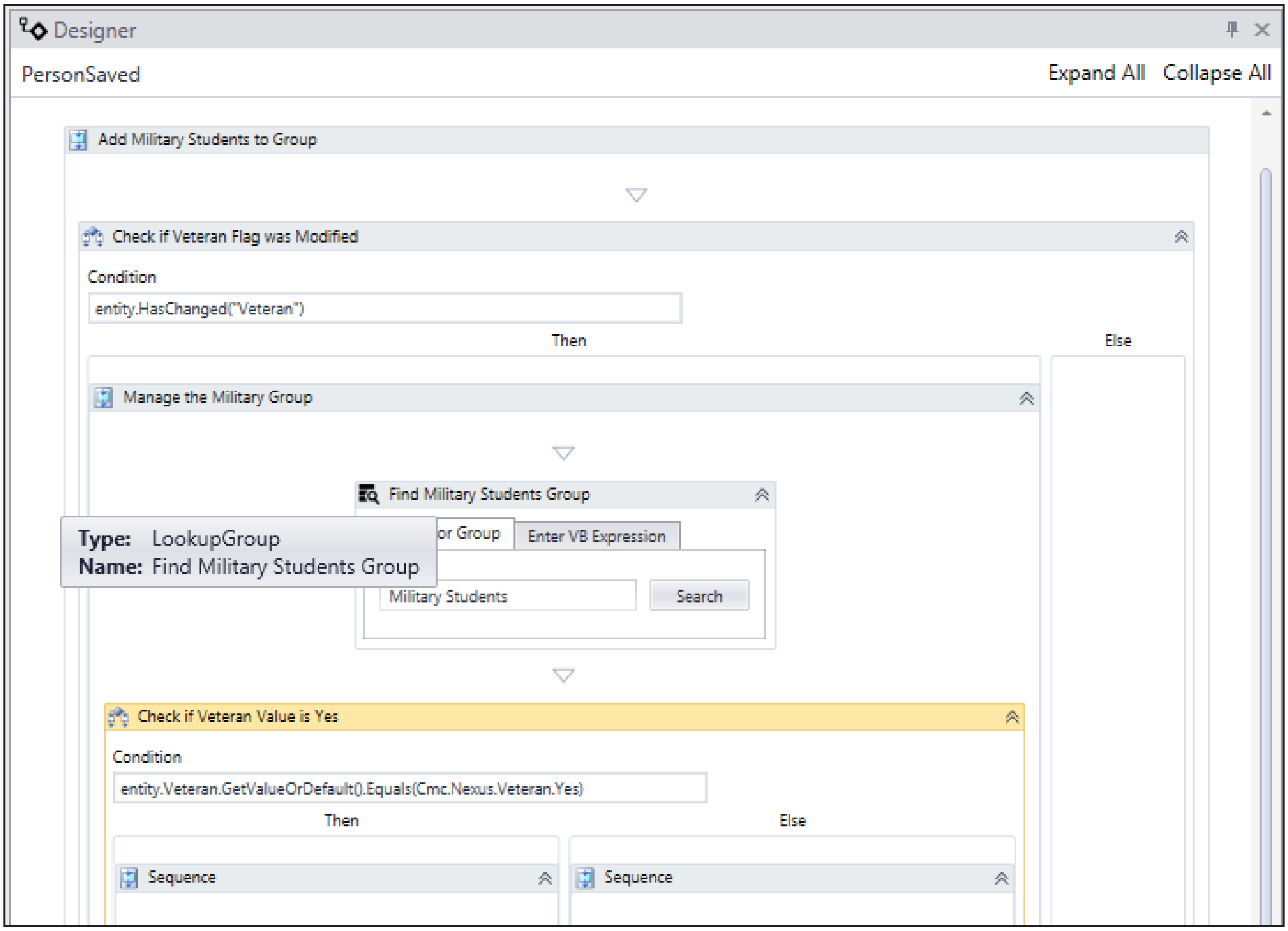
17. In Properties pane, specify **Check if the Veteran Value is Yes** as the DisplayName of the second If condition.

18. In the Condition field, specify the following VB expression:

entity.Veteran.GetValueOrDefault().Equals(Cmc.Nexus.Veteran.Yes)

The entity.Veteran.GetValueOrDefault() part of this expression gets the veteran status that was passed when the Veteran value was saved on the Person.

The Equals(Cmc.Nexus.Veteran.Yes) part of the expression calls the enumerated list of Veteran values in the Cmc.Nexus contract.


Designer
PersonSaved
Expand All
Collapse All
Add Military Students to Group
Check if Veteran Flag was Modified
Then
Else
Manage the Military Group
Find Military Students Group
Type: LookupGroup
Name: Find Military Students Group
or Group
Enter VB Expression
Search
Check if Veteran Value is Yes
Then
Else
Sequence
Sequence

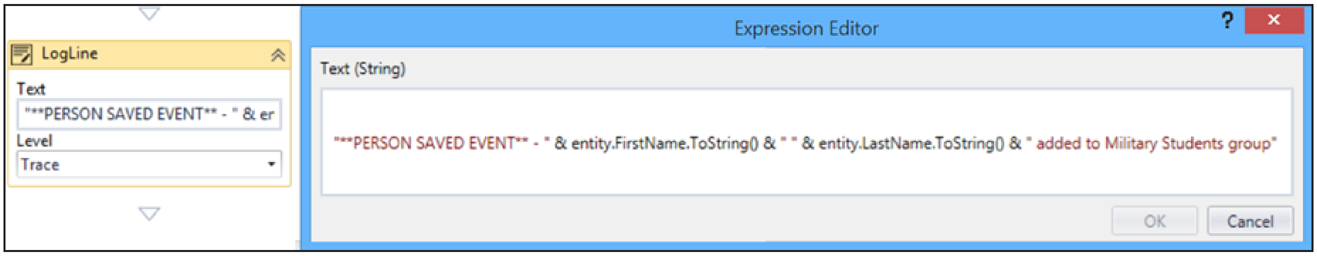
Condition: entity.HasChanged("Veteran")

Military Students

Condition: entity.Veteran.GetValueOrDefault().Equals(Cmc.Nexus.Veteran.Yes)

19. In the Toolbox, under Cmc.Core.Workflow.Activities, select the **LogLine** activity and drag it into the Then sequence of the second If condition.
20. Specify the following expression in the Text field of the LogLine activity:

"PERSON SAVED EVENT** - " & entity.FirstName.ToString() & " " & entity.LastName.ToString() & " added to Military Students group"**



Text
"**PERSON SAVED EVENT** - " & er

Level
Trace

Text (String)

"**PERSON SAVED EVENT** - " & entity.FirstName.ToString() & " " & entity.LastName.ToString() & " added to Military Students group"

21. In the Toolbox, under Cmc.Nexus.Workflow, select the **ManageGroupMembership** activity and drag it into the Then sequence of the second If condition.

22. In the Properties pane for the ManageGroupMembership activity, specify the following values:

    a. In the Action field, select **Add to Group**.

    b. In the DisplayName field, specify **Add to military students group**.

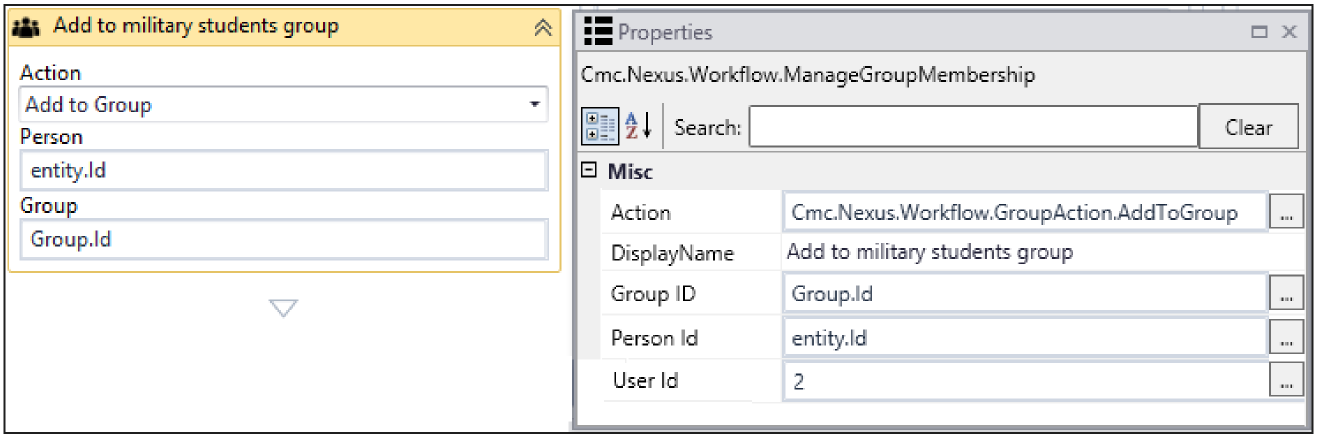
    c. In the Group field, specify **Group.Id**.

    d. In the Person field, specify **entity.Id**.

    e. In the User Id field, specify the User Id of the staff who is adding the group member.

The Group.Id is a variable from the LookupStudentGroup activity that will be used in the ManageGroupMembership activity.

The Add to Group action will only add the student to the group if the student is not already a group member.



Action
Add to Group

Person
entity.Id

Group
Group.Id

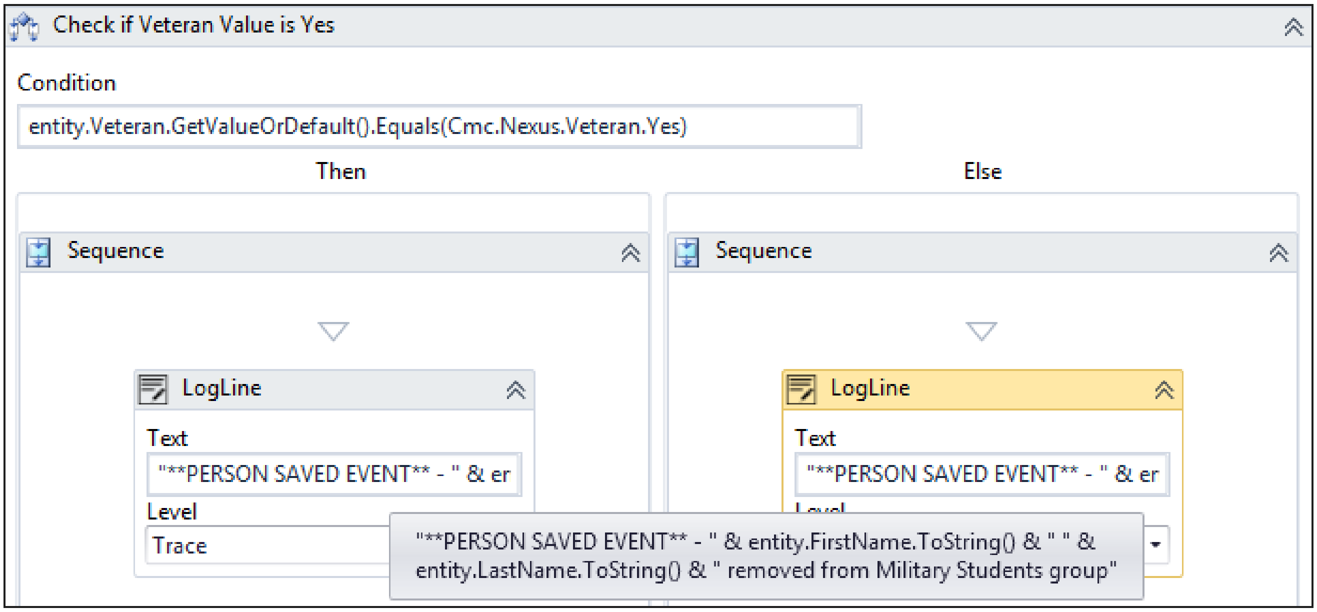
Cmc.Nexus.Workflow.ManageGroupMembership

Action	Cmc.Nexus.Workflow.GroupAction.AddToGroup
DisplayName	Add to military students group
Group ID	Group.Id
Person Id	entity.Id
User Id	2

23. Drag a **LogLine** activity into the Else sequence of the If condition named Check if the Veteran Value is Yes.

24. Specify the following expression in the Text field of the LogLine activity:

"PERSON SAVED EVENT** - " & entity.FirstName.ToString() & " " & entity.LastName.ToString() & " removed from Military Students group"**



Check if Veteran Value is Yes

Condition

entity.Veteran.GetValueOrDefault().Equals(Cmc.Nexus.Veteran.Yes)

Then

Sequence

LogLine

Text

"**PERSON SAVED EVENT** - " & er

Level

Trace

Else

Sequence

LogLine

Text

"**PERSON SAVED EVENT** - " & er

"**PERSON SAVED EVENT** - " & entity.FirstName.ToString() & " " & entity.LastName.ToString() & " removed from Military Students group"

25. Drag a **ManageGroupMembership** activity into the Else sequence of the If condition named Check if the Veteran Value is Yes.

26. In the Properties pane of the ManageGroupMembership activity, specify the following values:

    a. In the Action field, select **Remove from Group**.

    b. In the DisplayName field, specify **Remove from Military Students Group**.

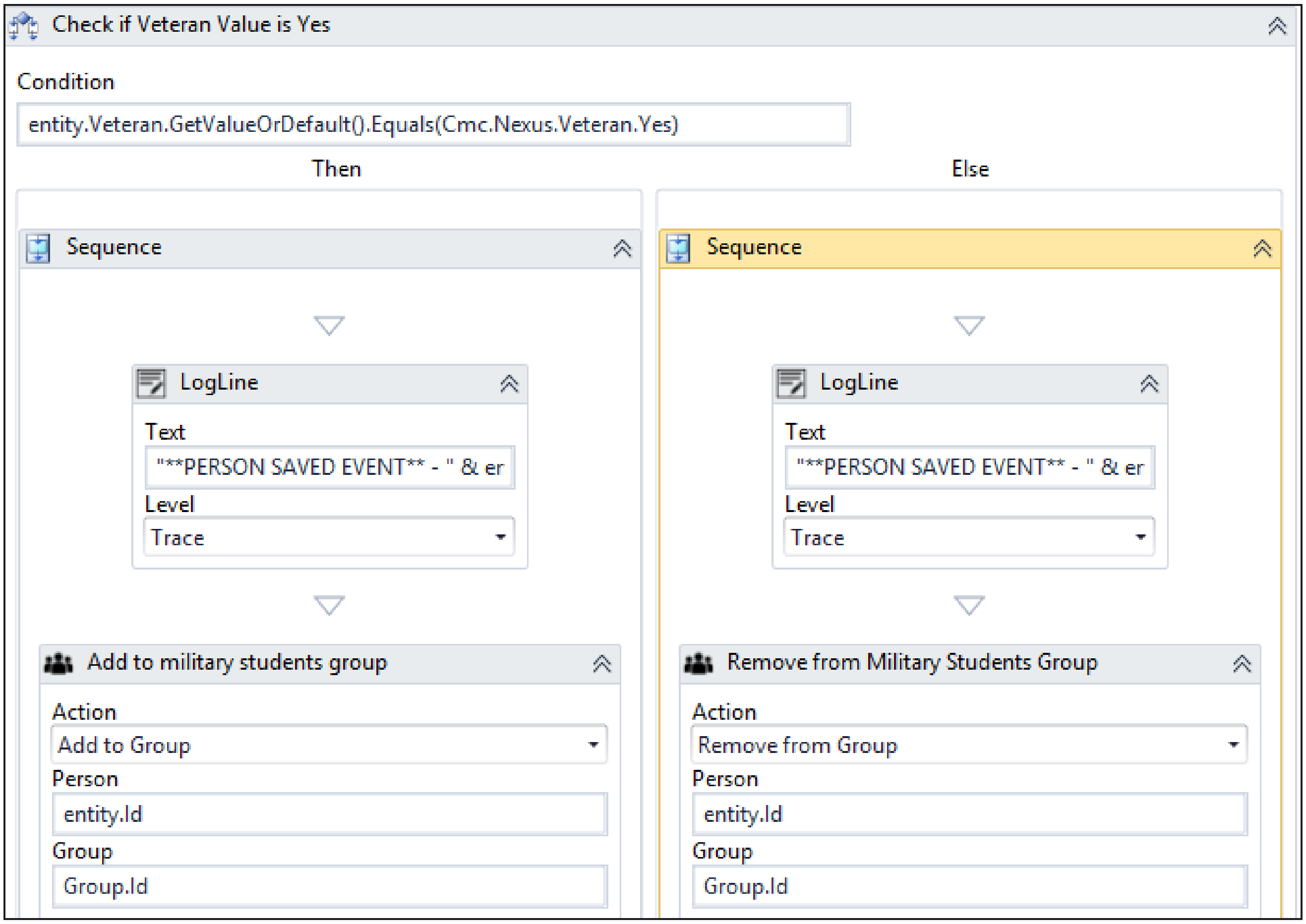
    c. In the Group field, specify **Group.Id**.

    d. In the Person field, specify **entity.Id**.

    e. In the User Id field, specify the User Id of the staff who is adding the group member.

    The Group.Id is a variable from the LookupStudentGroup activity that will be used in the ManageGroupMembership activity.

    The Remove from Group action will only remove the student from the group if the student is a group member.



Check if Veteran Value is Yes

Condition

entity.Veteran.GetValueOrDefault().Equals(Cmc.Nexus.Veteran.Yes)

Then

Sequence

LogLine

Text

"**PERSON SAVED EVENT** - " & er

Level

Trace

Add to military students group

Action

Add to Group

Person

entity.Id

Group

Group.Id

Else

Sequence

LogLine

Text

"**PERSON SAVED EVENT** - " & er

Level

Trace

Remove from Military Students Group

Action

Remove from Group

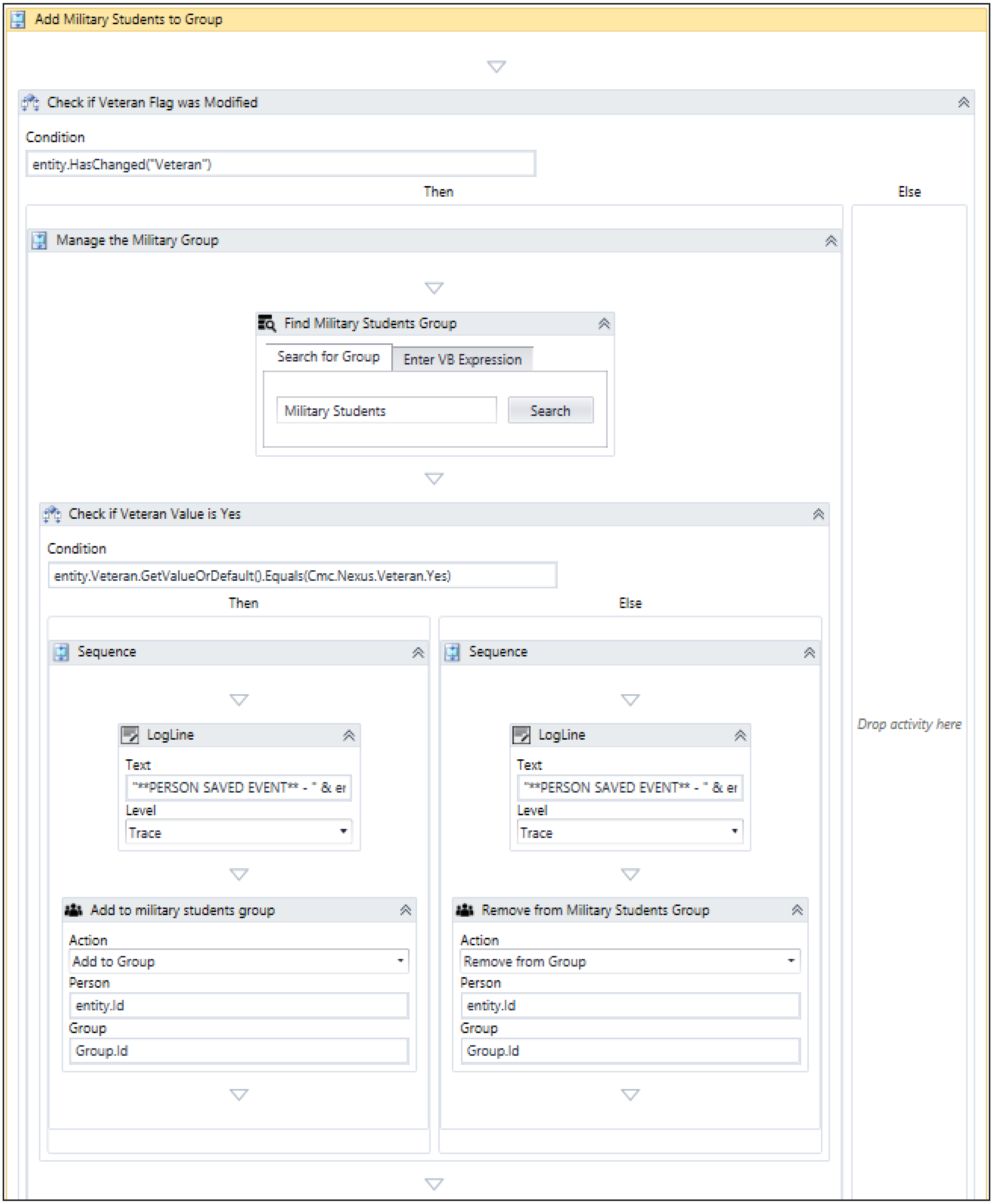
Person

entity.Id

Group

Group.Id

27. Check your workflow. Use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.



Add Military Students to Group

Check if Veteran Flag was Modified

Condition

entity.HasChanged("Veteran")

Then

Else

Manage the Military Group

Find Military Students Group

Military Students

Check if Veteran Value is Yes

Condition

entity.Veteran.GetValueOrDefault().Equals(Cmc.Nexus.Veteran.Yes)

Then

Else

Sequence

LogLine

Text

"**PERSON SAVED EVENT** - " & er

Level

Trace

Add to military students group

Action

Add to Group

Person

entity.Id

Group

Group.Id

Sequence

LogLine

Text

"**PERSON SAVED EVENT** - " & er

Level

Trace

Remove from Military Students Group

Action

Remove from Group

Person

entity.Id

Group

Group.Id

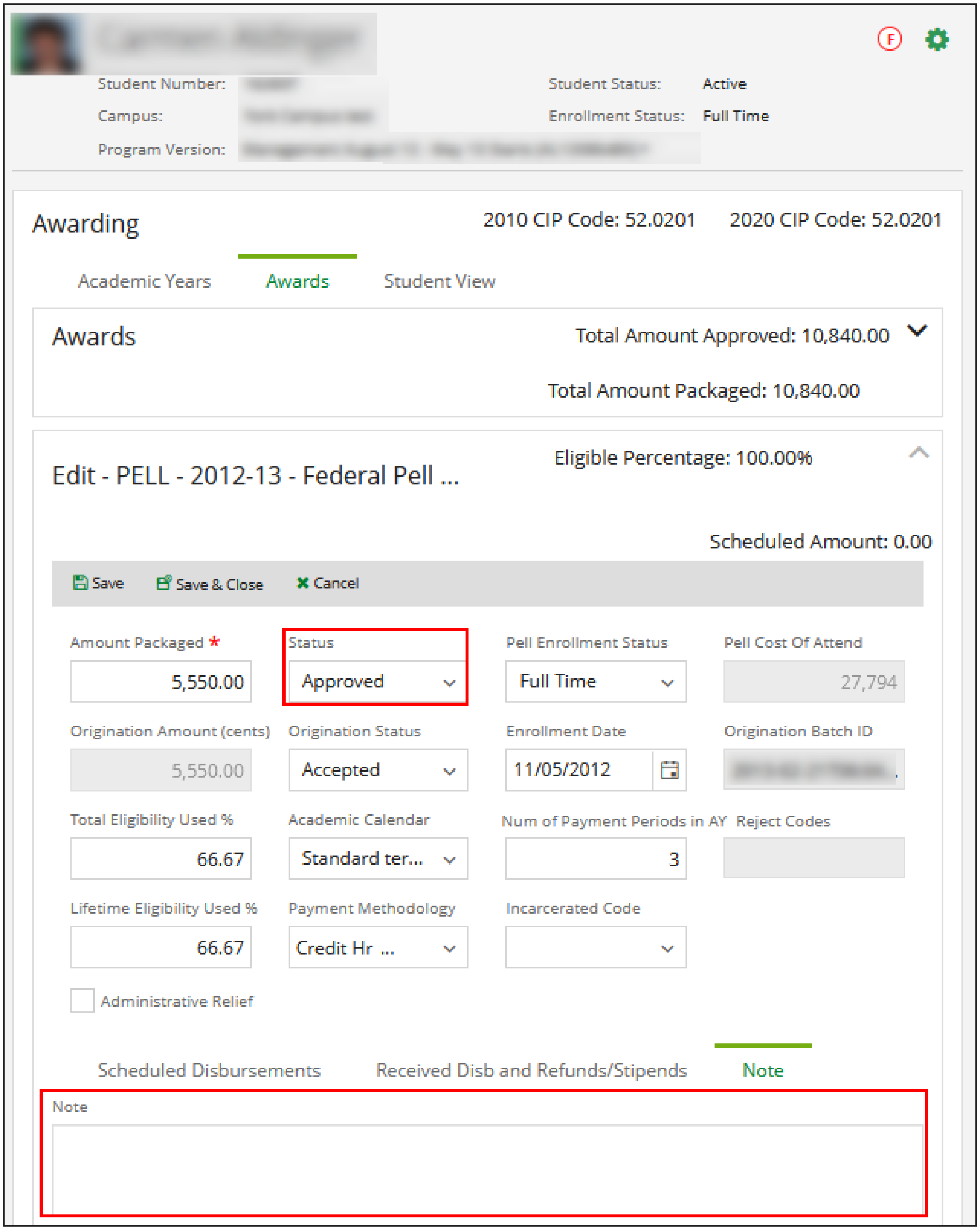
28. Click **Publish**. The New Workflow Definition Version window is displayed.

29. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.

30. Click **Save**, then **Cancel** to close the publisher window.

Check Approved Grants for Comments

This workflow checks for entries in the Note/Comments field when a grant or scholarship is saved with a status of 'Approved'. The workflow is triggered by a saving event that occurs when the Save button is clicked on the Awards form (standard interface) or on the Financial Aid Grant / Scholarship form (legacy interface).



[redacted]

Student Number: [redacted]
Campus: [redacted]
Program Version: [redacted]

Student Status: Active
Enrollment Status: Full Time

2010 CIP Code: 52.0201 2020 CIP Code: 52.0201

Total Amount Approved: 10,840.00

Total Amount Packaged: 10,840.00

Edit - PELL - 2012-13 - Federal Pell ...

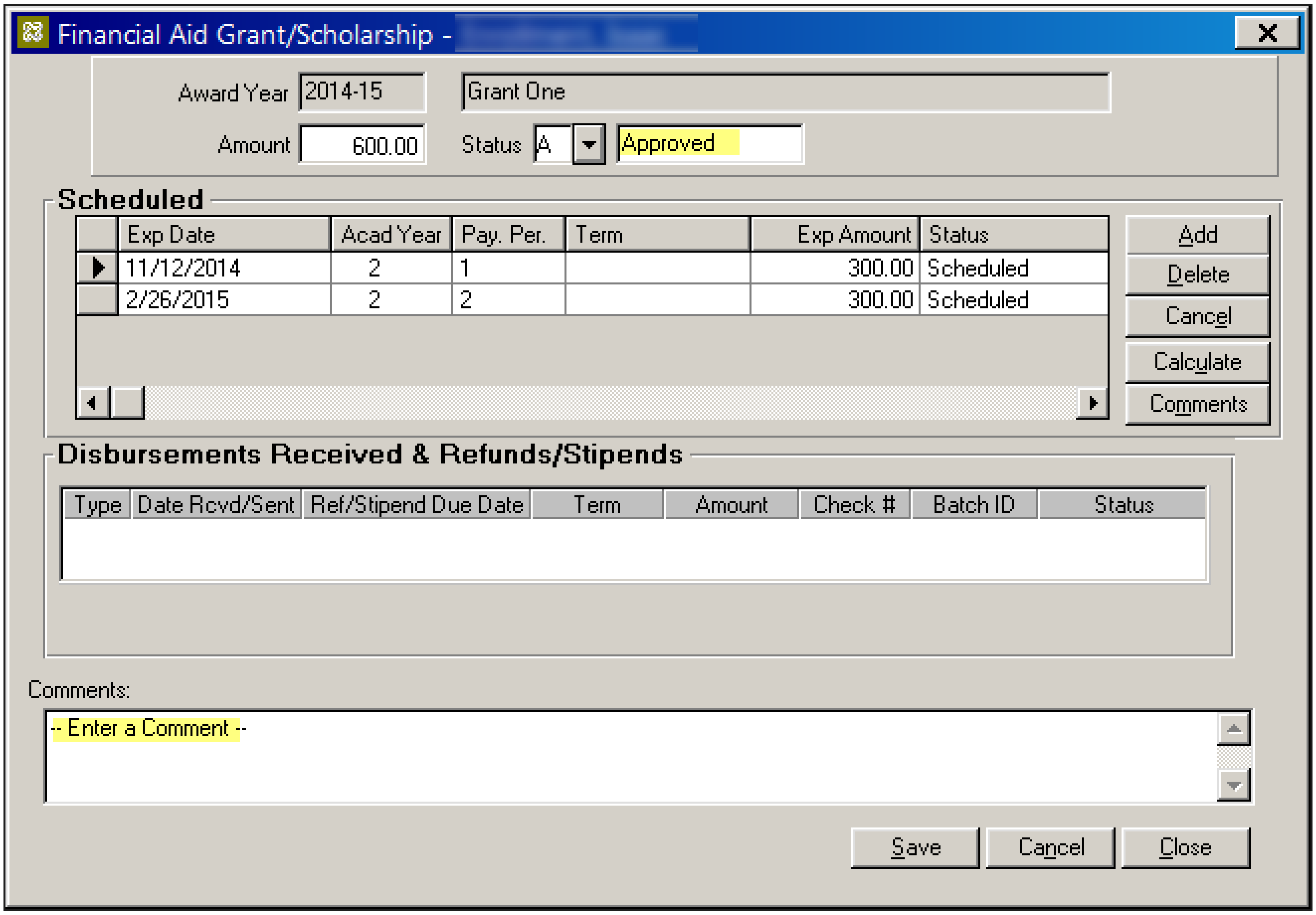
Eligible Percentage: 100.00%

Scheduled Amount: 0.00

Amount Packaged *	Status	Pell Enrollment Status	Pell Cost Of Attend
5,550.00	Approved	Full Time	27,794
Origination Amount (cents)	Origination Status	Enrollment Date	Origination Batch ID
5,550.00	Accepted	11/05/2012	[redacted]
Total Eligibility Used %	Academic Calendar	Num of Payment Periods in AY	Reject Codes
66.67	Standard ter...	3	
Lifetime Eligibility Used %	Payment Methodology	Incarcerated Code	
66.67	Credit Hr ...		

□ Administrative Relief

Note



Financial Aid Grant/Scholarship - [redacted]

Award Year 2014-15 Grant One

Amount 600.00 Status A Approved

Scheduled

	Exp Date	Acad Year	Pay. Per.	Term	Exp Amount	Status
▶	11/12/2014	2	1		300.00	Scheduled
	2/26/2015	2	2		300.00	Scheduled

Disbursements Received & Refunds/Stipends

Type	Date Rcvd/Sent	Ref/Stipend Due Date	Term	Amount	Check #	Batch ID	Status

Comments:

-- Enter a Comment --

1. Start the **Workflow** application from your desktop.

2. On the Home tab, click **New Event Workflow**.

3. In the Entities area:

   a. Click ▶ next to **Cmc.Nexus.Contracts**.

   b. Click ▶ next to **Cmc.Nexus.Sis.FinancialAid**.

   c. Click **Student Grant Detail (StudentAwardDetailGrant)**.

4. In the Events area, click **Saving (SavingEvent)**.

5. Specify a **Name** for the workflow and click **OK**.

6. Drag an **If** activity it into the sequence.

   a. In the Condition field, specify the following expression:

   (String.IsNullOrEmpty(entity.Note)) AND (entity.Status.Equals("Approved"))

7. Drag a **CreateValidationItem** activity into the Then branch of the If condition.

   a. In the Message field, specify the following string:

      "If grant status is Approved, then a comment is required."

      This message will be displayed in Anthology Student when an approved grant is saved without a comment.

   b. In the Message Type field, select **Error** (default).

   c. In the Messages field of the Properties pane, enter **args.ValidationMessages**.

8. Drag a **LogLine** activity into the Else branch of the If condition.

   a. In the Text field, specify the following expression:

      "Grant condition check false" & Environment.NewLine

      This expression creates a new line in the event log with the text "Grant condition check false".

   b. In the Level field, select **Information** (default).

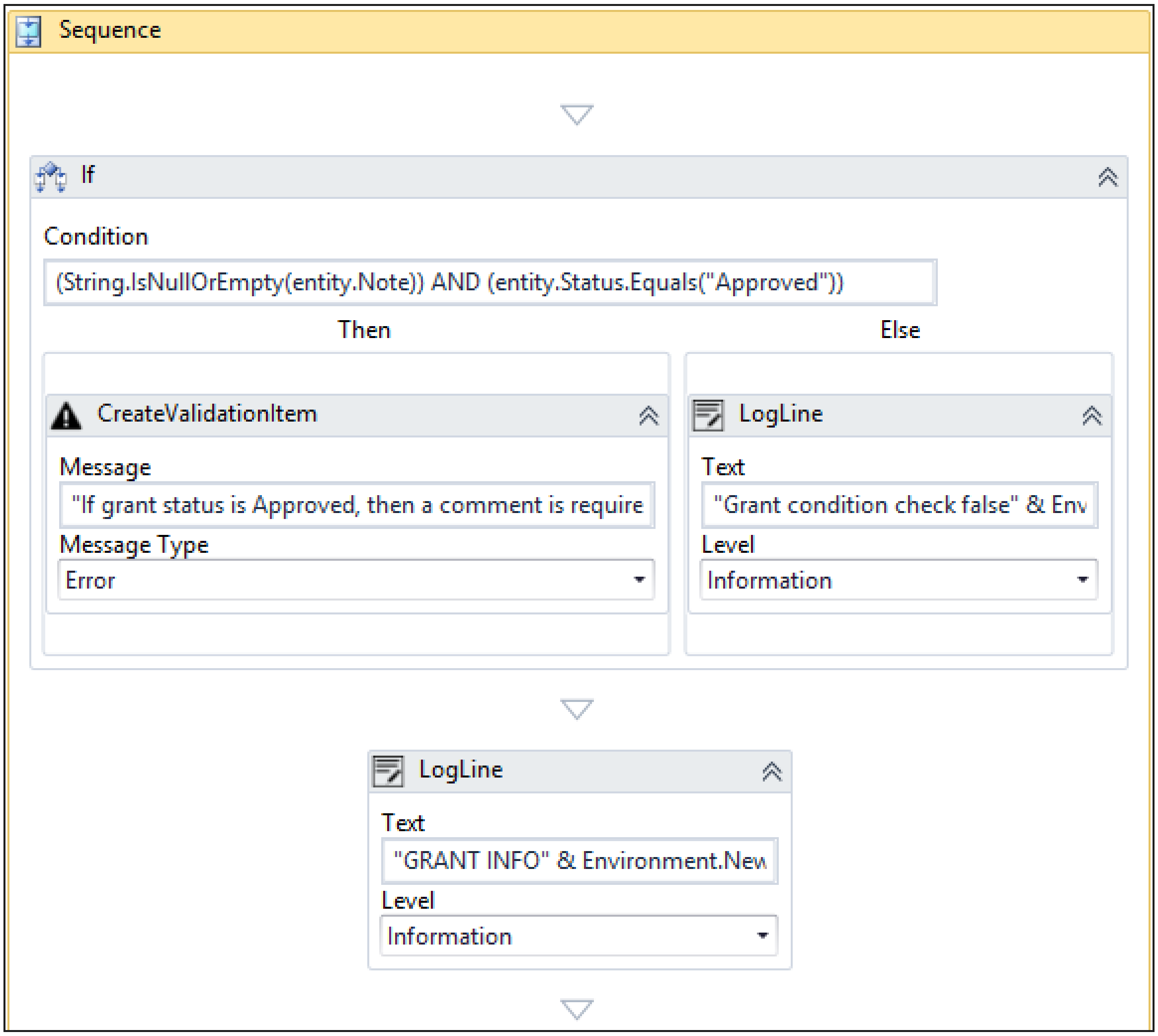
9. Drag a **LogLine** activity into the sequence below the If condition.

   a. In the Text field, specify the following expression:

      "GRANT INFO" & Environment.NewLine & " Award Amount: " & entity.AwardAmount & Environment.NewLine & " Create Date: " & entity.CreateDate & Environment.NewLine & " CreatedByUserId: " & entity.CreatedByUserId & Environment.NewLine & " Fund Source ID: " & entity.FundSourceId & Environment.NewLine & " ID: " & entity.Id & Environment.NewLine & " Modified By User ID: " & entity.ModifiedByUserId & Environment.NewLine & " Note: " & entity.Note & Environment.NewLine & " Status: " & entity.Status & Environment.NewLine & " Student Academic Year ID: " & entity.StudentAcademicYearId & Environment.NewLine & " Student Award Summary ID: " & entity.StudentAwardSummaryId

      This expression captures the data from the top section of the Financial Aid Grant / Scholarship form in the event log.

   b. In the Level field, select **Information** (default).



(String.IsNullOrEmpty(entity.Note)) AND (entity.Status.Equals("Approved"))

Message: "If grant status is Approved, then a comment is require
Message Type: Error

Text: "Grant condition check false" & Env
Level: Information

Text: "GRANT INFO" & Environment.New
Level: Information

10. Drag another **If** activity into the sequence.

    a. In the Condition field, specify the following expression:

       entity.ScheduledDisbursements.Count > 0

11. Drag a **ForEach** activity into the Then branch of the If condition.

    a. In the Foreach item in field, specify the following expression:

       entity.ScheduledDisbursements

    b. Drag a **LogLine** activity into the Body of the ForEach activity.

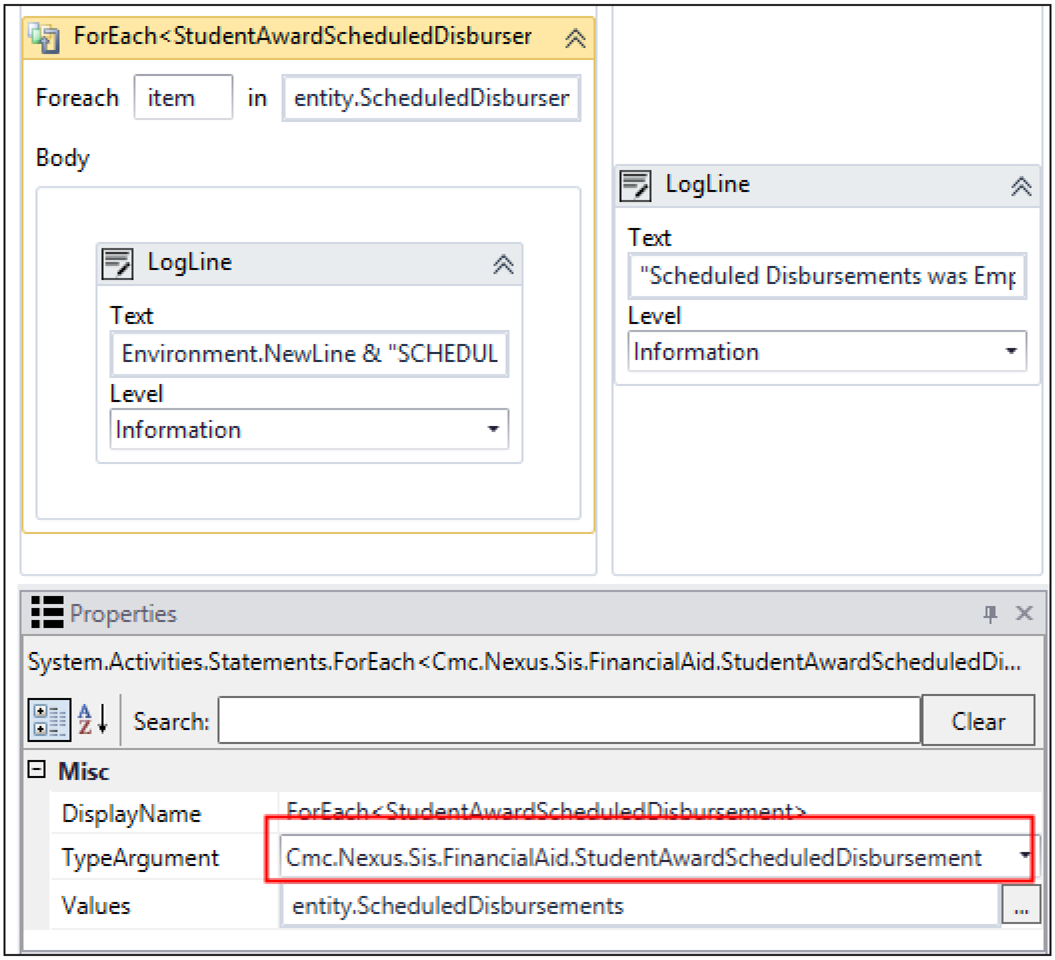
    c. In the Text field of the LogLine activity, specify the following expression:

       Environment.NewLine & "SCHEDULED DISBURSEMENT LINE ITEM: " & Environment.NewLine & " Amount Expected: " & item.AmountExpected & Environment.NewLine & " Disbursement

Number: " & item.DisbursementNumber & Environment.NewLine & " ExpectedDate: " & item.ExpectedDate & Environment.NewLine & " ID: " & item.Id & Environment.NewLine & " Lender Fee: " & item.LenderFee & Environment.NewLine & " Note: " & item.Note & Environment.NewLine & " Status: " & item.Status & Environment.NewLine & " StudAcadYearPP Id: " & item.StudentAcademicYearPaymentPeriod.Id & Environment.NewLine & " StudAcadYearPP PayPer Description: " & item.StudentAcademicYearPaymentPeriod.PaymentPeriod.Description & Environment.NewLine & " StudAcadYearPP PayPer Id: " & item.StudentAcademicYearPaymentPeriod.PaymentPeriod.Id & Environment.NewLine & " StudAcadYearPP PayPer TermId: " & item.StudentAcademicYearPaymentPeriod.PaymentPeriod.TermId & Environment.NewLine & " StudAcadYearPP EndDate: " & item.StudentAcademicYearPaymentPeriod.PaymentPeriodEndDate & Environment.NewLine & " StudAcadYearPP StartDate: " & item.StudentAcademicYearPaymentPeriod.PaymentPeriodStartDate & Environment.NewLine & " StudAcadYearPP Sequence: " & item.StudentAcademicYearPaymentPeriod.Sequence & Environment.NewLine & " StudAcadYearPP StudAcadYearId: " & item.StudentAcademicYearPaymentPeriod.StudentAcademicYearId

This expression captures the data from the Scheduled Disbursements section of the Financial Aid Grant / Scholarship form in the event log.

d. In the Properties pane of the ForEach activity, specify the following object type in the TypeArgument field: **Cmc.Nexus.Sis.FinancialAid.StudentAwardScheduledDisbursement**.



Foreach item in entity.ScheduledDisburser

LogLine

Text: Environment.NewLine & "SCHEDUL

Level: Information

LogLine

Text: "Scheduled Disbursements was Emp

Level: Information

System.Activities.Statements.ForEach<Cmc.Nexus.Sis.FinancialAid.StudentAwardScheduledDi...

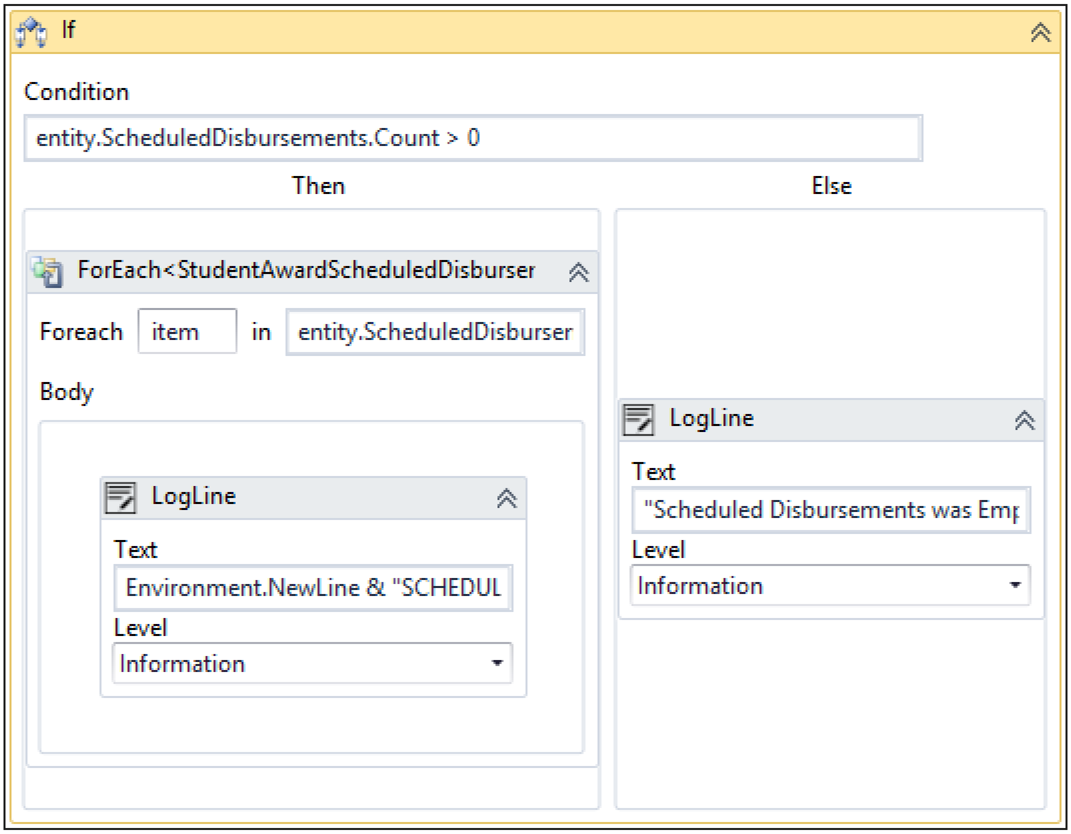
DisplayName	ForEach<StudentAwardScheduledDisbursement>
TypeArgument	Cmc.Nexus.Sis.FinancialAid.StudentAwardScheduledDisbursement
Values	entity.ScheduledDisbursements

12. Drag a **LogLine** activity into the Else branch of the If condition.

    a. In the Text field, specify the following expression:

    "Scheduled Disbursements was Empty" & Environment.NewLine

    This expression creates a new line in the event log with the text "Scheduled Disbursements was Empty".



Condition: entity.ScheduledDisbursements.Count > 0

Foreach item in entity.ScheduledDisburser

Text: Environment.NewLine & "SCHEDUL

Level: Information

Text: "Scheduled Disbursements was Emp

Level: Information

13. Check your workflow. Scroll through the workflow or use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.

14. Click **Publish**. The New Workflow Definition Version window is displayed.

15. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.

16. Click **Save**, then **Cancel** to close the publisher window.

Check if a Grade was Posted

This workflow checks if a grade was posted for a student who did not attend classes. If a grade was posted, a task is created to inform the student's advisor.

1. Start the **Workflow** application from your desktop.
2. On the Home tab, click **New Event Workflow**.
3. In the Entities area:
    a. Click ▶ next to **Cmc.Nexus.Contracts**.
    b. Click ▶ next to **Cmc.Nexus.Sis.Academics**.
    c. Click **Student Course (StudentCourse)**.
4. In the Events area, click **Saved (SavedEvent)**.
5. Specify a **Name** for the workflow and click **OK**.
6. Drag an **If** activity it into the sequence.
    a. In the Condition field, specify the following expression:

       entity.HasChanged("GradePostedDate")

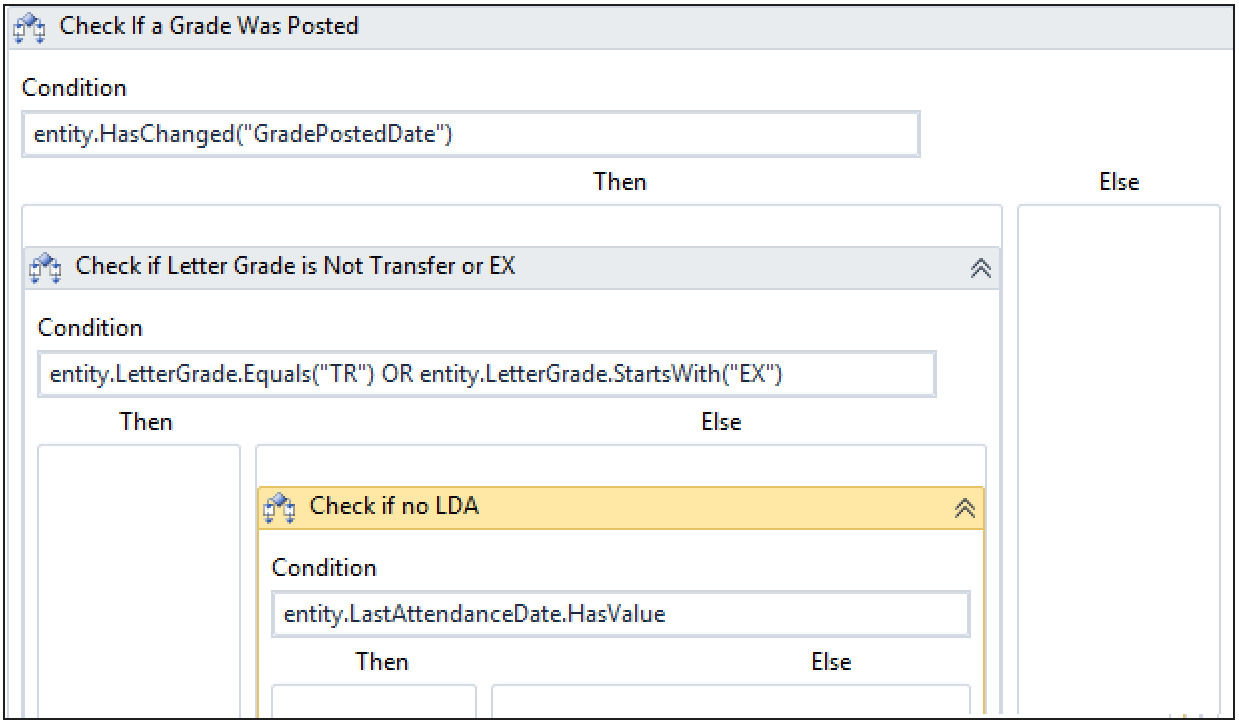
    b. In the DisplayName property field, specify **Check If a Grade Was Posted**.
7. Drag an **If** activity into the Then branch of the first If condition.
    a. In the Condition field, specify the following expression:

       entity.LetterGrade.Equals("TR") OR entity.LetterGrade.StartsWith("EX")

    b. In the DisplayName property field, specify **Check if Letter Grade is Not Transfer or EX**.
8. Drag an **If** activity into the Else branch of the second If condition.
    a. In the Condition field, specify the following expression:

       entity.LastAttendanceDate.HasValue

    b. In the DisplayName property field, specify **Check if no LDA**.



Check If a Grade Was Posted

Condition

entity.HasChanged("GradePostedDate")

Check if Letter Grade is Not Transfer or EX

Condition

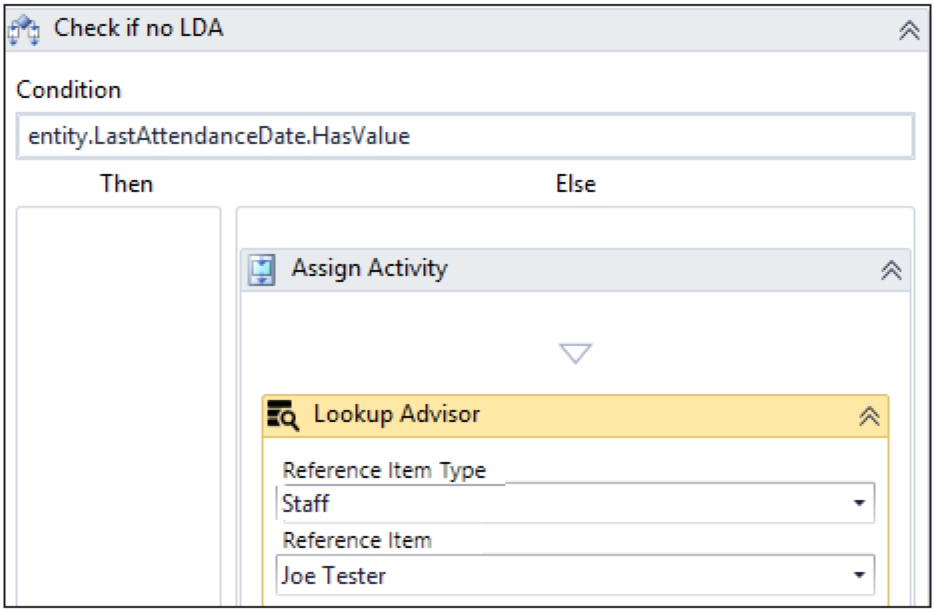
entity.LetterGrade.Equals("TR") OR entity.LetterGrade.StartsWith("EX")

Check if no LDA

Condition

entity.LastAttendanceDate.HasValue

9. Drag a **Sequence** activity into the Else branch of third If condition.
   a. In the DisplayName property field, specify **Assign Activity**.
10. Drag a **LookupReferenceItem** activity into the Assign Activity sequence.
    a. In the Reference Item Type field, select **Staff**.
    b. In the Reference Item field, select an advisor.
    c. In the DisplayName property field, specify **Lookup Advisor**.
    d. In the Variables pane, create a variable to hold the **advisor** that was looked up.



Check if no LDA

Condition: entity.LastAttendanceDate.HasValue

Then

Else

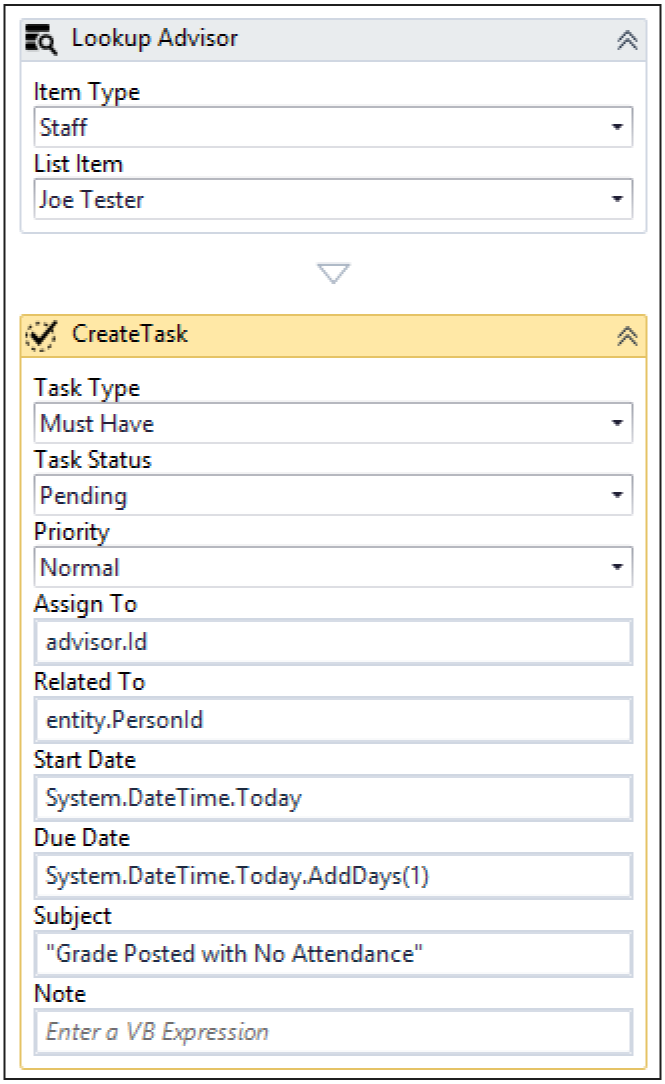
Assign Activity

Lookup Advisor

Reference Item Type: Staff

Reference Item: Joe Tester

11. Drag a **Create Task** activity into the Else branch of the If condition.
    a. In the Task Type field, select **Must Have**.
    b. In the Task Status field, select **Pending**.
    c. In the Priority field, select **Normal**.
    d. In the Assign To field, specify **advisor.Id**.
    e. In the Related To field, specify **entity.PersonId**.
    f. In the Start Date field, specify **System.DateTime.Today**.
    g. In the DueTo field, specify **System.DateTime.Today.AddDays(1)**.
    h. In the Subject field, specify **"Grade Posted with No Attendance"**.


Lookup Advisor

Item Type: Staff

List Item: Joe Tester

CreateTask

Task Type: Must Have

Task Status: Pending

Priority: Normal

Assign To: advisor.Id

Related To: entity.PersonId

Start Date: System.DateTime.Today

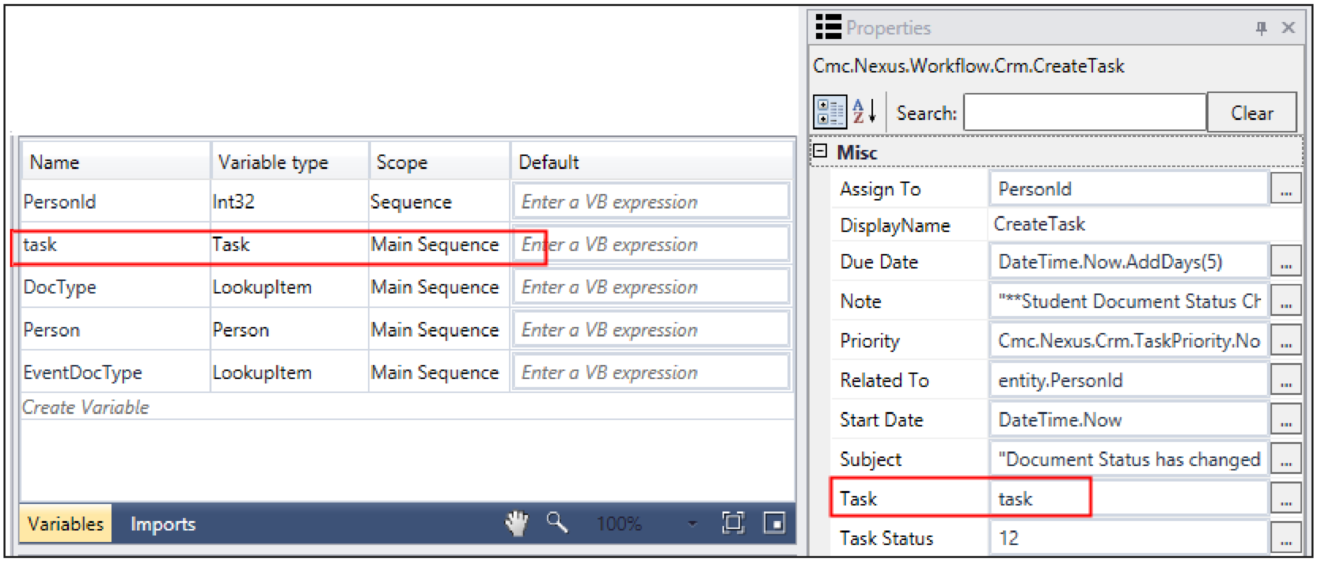
Due Date: System.DateTime.Today.AddDays(1)

Subject: "Grade Posted with No Attendance"

Note: *Enter a VB Expression*

12. In the Variables pane, create a variable that holds the **Task** instance object.

    Enter the name of the variable in the Task field of the Properties pane for the Create Task activity.



Name	Variable type	Scope	Default
PersonId	Int32	Sequence	Enter a VB expression
task	Task	Main Sequence	Enter a VB expression
DocType	LookupItem	Main Sequence	Enter a VB expression
Person	Person	Main Sequence	Enter a VB expression
EventDocType	LookupItem	Main Sequence	Enter a VB expression
Create Variable			

Cmc.Nexus.Workflow.Crm.CreateTask

Misc

Assign To	PersonId
DisplayName	CreateTask
Due Date	DateTime.Now.AddDays(5)
Note	"**Student Document Status Ch
Priority	Cmc.Nexus.Crm.TaskPriority.No
Related To	entity.PersonId
Start Date	DateTime.Now
Subject	"Document Status has changed
Task	task
Task Status	12

13. Drag a **Save Task** activity into the sequence below the CreateTask activity.

    Enter the name of the variable that holds the **Task** instance object in the Task field of the Properties pane for the Save Task activity.

14. Check your workflow. Scroll through the workflow or use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.

15. Click **Publish**. The New Workflow Definition Version window is displayed.

16. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.

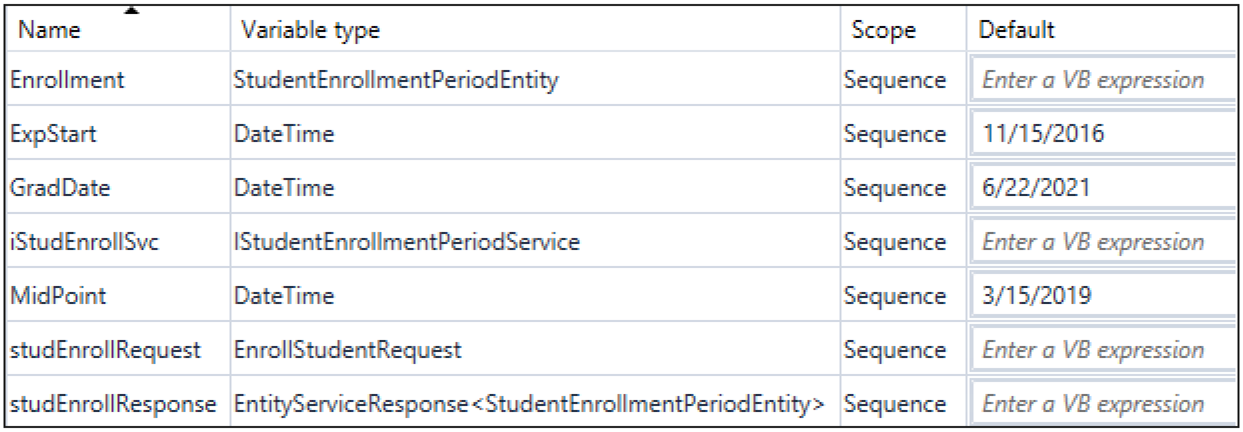
17. Click **Save**, then **Cancel** to close the publisher window.

Create a Student Enrollment Period

This workflow creates a new student enrollment period and assigns an enrollment number to a student. The workflow uses the GetServiceInstance<IStudentEnrollmentPeriodService> activity. This service creates a new enrollment record with a proper student enrollment number. The service calls the InsertStudentEnrollment method on the iStudentEnrollmentPeriodService to insert a new enrollment record in the AdEnroll table.

Note: Do not use the SaveEntity<StudentEnrollmentEntity> activity to create a new student enrollment period record. The saved record will not have a new enrollment number in the StuNum field of the AdEnroll table.

1. Open the workflow where you want to create an enrollment.
2. Create the following **variables**. Specify default values as needed for your use case.



Name	Variable type	Scope	Default
Enrollment	StudentEnrollmentPeriodEntity	Sequence	*Enter a VB expression*
ExpStart	DateTime	Sequence	11/15/2016
GradDate	DateTime	Sequence	6/22/2021
iStudEnrollSvc	IStudentEnrollmentPeriodService	Sequence	*Enter a VB expression*
MidPoint	DateTime	Sequence	3/15/2019
studEnrollRequest	EnrollStudentRequest	Sequence	*Enter a VB expression*
studEnrollResponse	EntityServiceResponse<StudentEnrollmentPeriodEntity>	Sequence	*Enter a VB expression*

3. Drag a **CreateEntity** activity into the sequence.
   a. In the TEntity field, select **Browse for Types...**.
   b. Select **Cmc.Nexus.Academics.Contracts** > **Cmc.Nexus.Academics.Entities** > **StudentEnrollmentPeriodEntity** and click **OK**.
   c. In the Result property field, specify the **Enrollment** variable created above.
4. Drag an **Assign** activity into the sequence.
   a. In the "To" field specify **studEnrollRequest** (variable created above).
   b. In the "Value" field specify **new EnrollStudentRequest(Enrollment)**.
5. Drag a **GetServiceInstance** activity into the workflow.
   a. In the TService field, select **Browse for Types...**
   b. Select **Cmc.Nexus.Academics.Services** > **IStudentEnrollmentPeriodService** and click **OK**.
   c. In the Result property field, specify the **iStudEnrollSvc** variable created above.
6. Insert **Assign** activities for each row in the following table. Assign values as needed for your environment.

Note: Use LookupReferenceItem or other methods to find the correct values for these fields. They have been hard-coded in a test environment for example purposes only.

Assign Properties

To	Value	Value Example
studEnrollRequest.entity.StudentId	<StudentID>	20073
studEnrollRequest.entity.SchoolStatusId	<SchoolStatusId>	5
studEnrollRequest.entity.CampusId	<CampusId>	1
studEnrollRequest.entity.ProgramId	<ProgramId>	83
studEnrollRequest.entity.ProgramVersionId	<ProgramVersionId>	166
studEnrollRequest.entity.ShiftId	<ShiftId>	64
studEnrollRequest.entity.BillingMethodId	<BillingMethodId>	52
studEnrollRequest.entity.GradeLevelId	<GradeLevelId>	1
studEnrollRequest.entity.StartDateId	<StartDateId>	2998
studEnrollRequest.entity.IpedsState	<IpedsState>	"GA"
studEnrollRequest.entity.AcademicAdvisorId	<AcademicAdvisorId>	115
studEnrollRequest.entity.ApplicationReceivedDate	<ApplicationReceivedDate>	datetime.Today
studEnrollRequest.entity.EnrollmentDate	<EnrollmentDate>	datetime.Today
studEnrollRequest.entity.MidpointDate	<MidpointDate>	MidPoint (variable created above)
studEnrollRequest.entity.GraduationDate	<GraduationDate>	GradDate (variable created above)
studEnrollRequest.entity.ExpectedStartDate	<ExpectedStartDate>	ExpStart (variable created above)

7. Optionally, for testing purposes, insert a **WriteLine** activity with Text = "Before Call".

8. Drag an **Assign** activity into the sequence. This is the call to the GetServiceInstance activity using variables created above.

   a. In the "To" field specify **studEnrollResponse**.

   b. In the "Value" field specify **iStudEnrollSvc.InsertStudentEnrollment(studEnrollRequest)**.

9. Optionally, insert another **WriteLine** with Text = "After Call".

10. Optionally, insert a **LogObject** activity with Level = Error and Object = studEnrollResponse.

11. Click **Publish**. The New Workflow Definition Version window is displayed.

12. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.

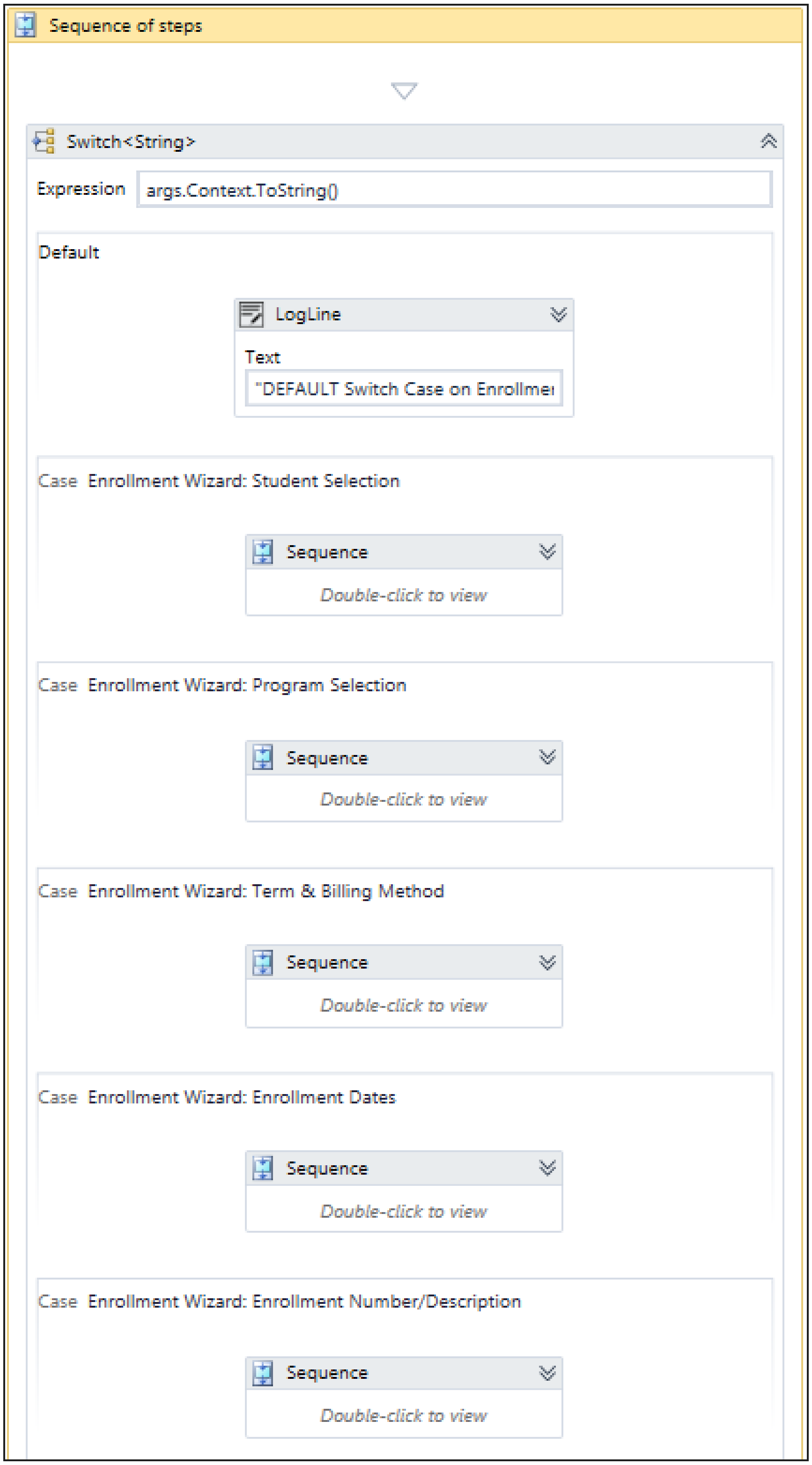
13. Click **Save**, then **Cancel** to close the publisher window.

Custom Field Validations on Each Step of Enrollment Wizard

This workflow performs custom validations on each step of the Anthology Student Enrollment wizard. This workflow example is described at a high level using screen captures for the main sequences, rather than describing each step in detail. The workflow is triggered by a Saving event using the contract Cmc.Nexus.Contracts > Cmc.Nexus > Person.

Note: When the Student Enrollment wizard uses a Person Saving event, each step only fills out a few fields in the Person.Students(0).StudentEnrollmentPeriods(0) entity based on the step Context.

The top level sequence contains a Switch activity based on the args.Context for each step of the enrollment wizard, plus a default case.



Sequence of steps

Switch<String>

Expression args.Context.ToString()

Default

LogLine

Text "DEFAULT Switch Case on Enrollmei

Case Enrollment Wizard: Student Selection

Sequence

Double-click to view

Case Enrollment Wizard: Program Selection

Sequence

Double-click to view

Case Enrollment Wizard: Term & Billing Method

Sequence

Double-click to view

Case Enrollment Wizard: Enrollment Dates

Sequence

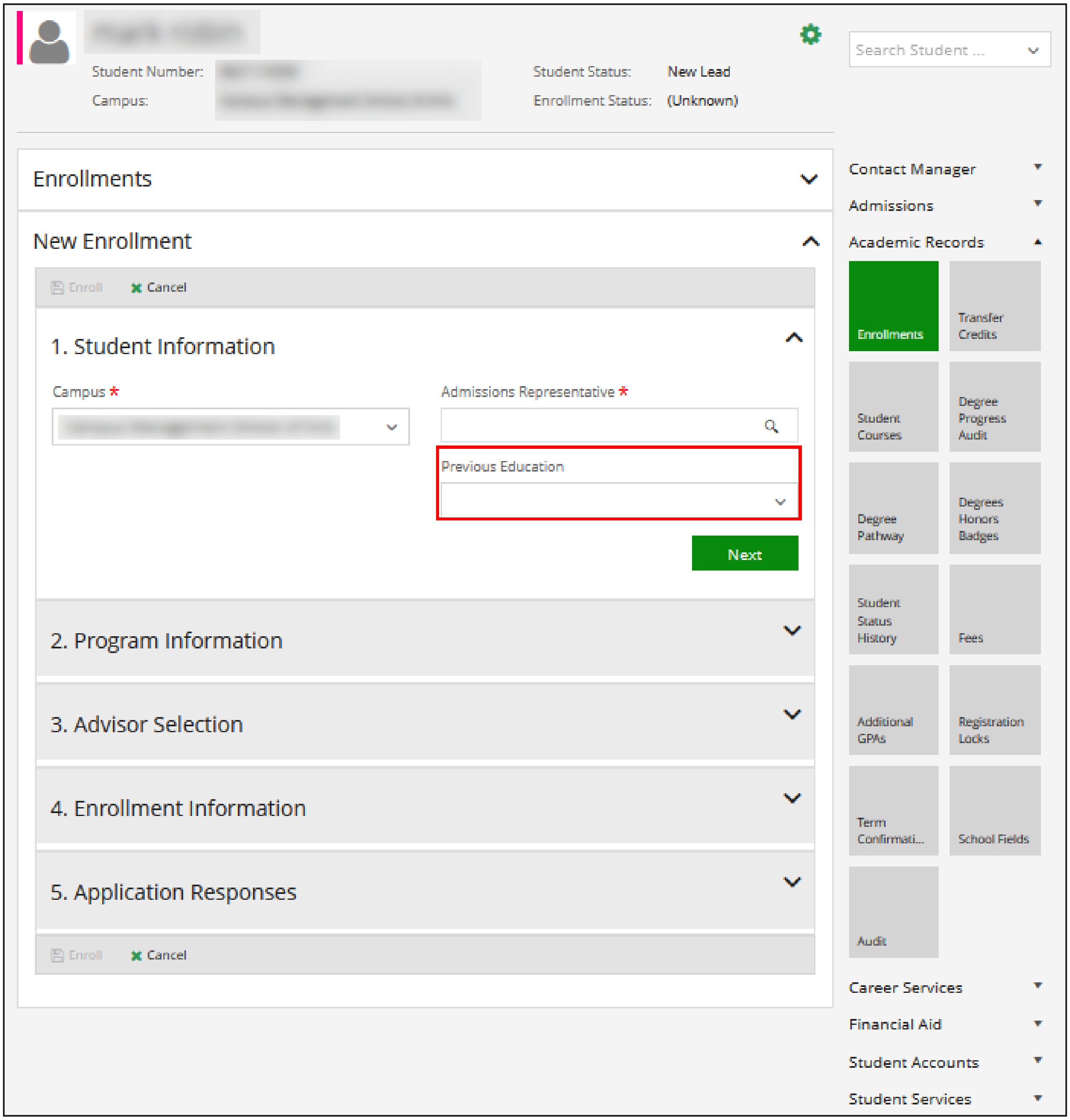
Double-click to view

Case Enrollment Wizard: Enrollment Number/Description

Sequence

Double-click to view

1. The first case creates an error message when the value "Homeschooled" is selected in the **Previous Education** field of the Enrollment wizard.



[redacted]

Student Number: [redacted]

Campus: [redacted]

Student Status: New Lead

Enrollment Status: (Unknown)

Enrollments

New Enrollment

1. Student Information

Campus *: [redacted]

Admissions Representative *:

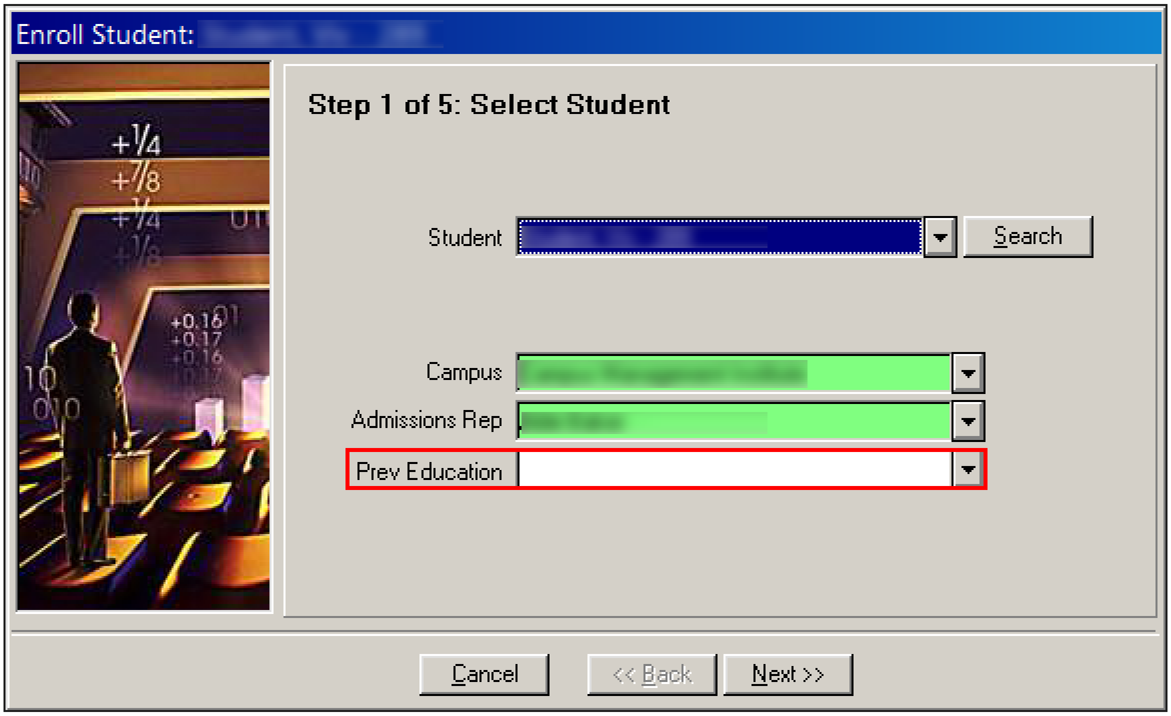
Previous Education:

2. Program Information

3. Advisor Selection

4. Enrollment Information

5. Application Responses



Enroll Student: [redacted]

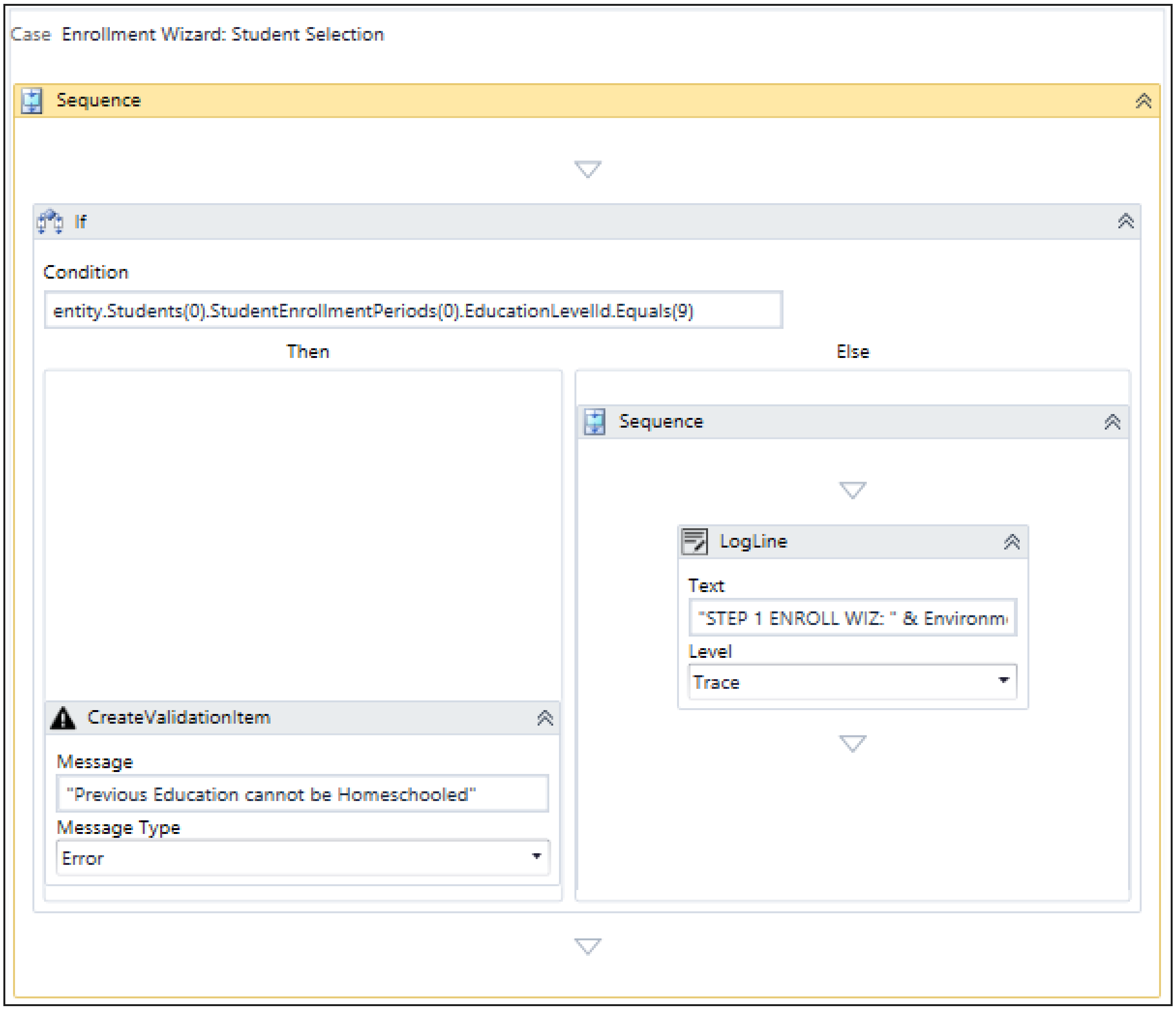
Step 1 of 5: Select Student

Student [redacted]

Campus [redacted]

Admissions Rep [redacted]

Prev Education



Case Enrollment Wizard: Student Selection

Sequence

If

Condition

```
entity.Students(0).StudentEnrollmentPeriods(0).EducationLevelId.Equals(9)
```

Then

CreateValidationItem

Message: "Previous Education cannot be Homeschooled"

Message Type: Error

Else

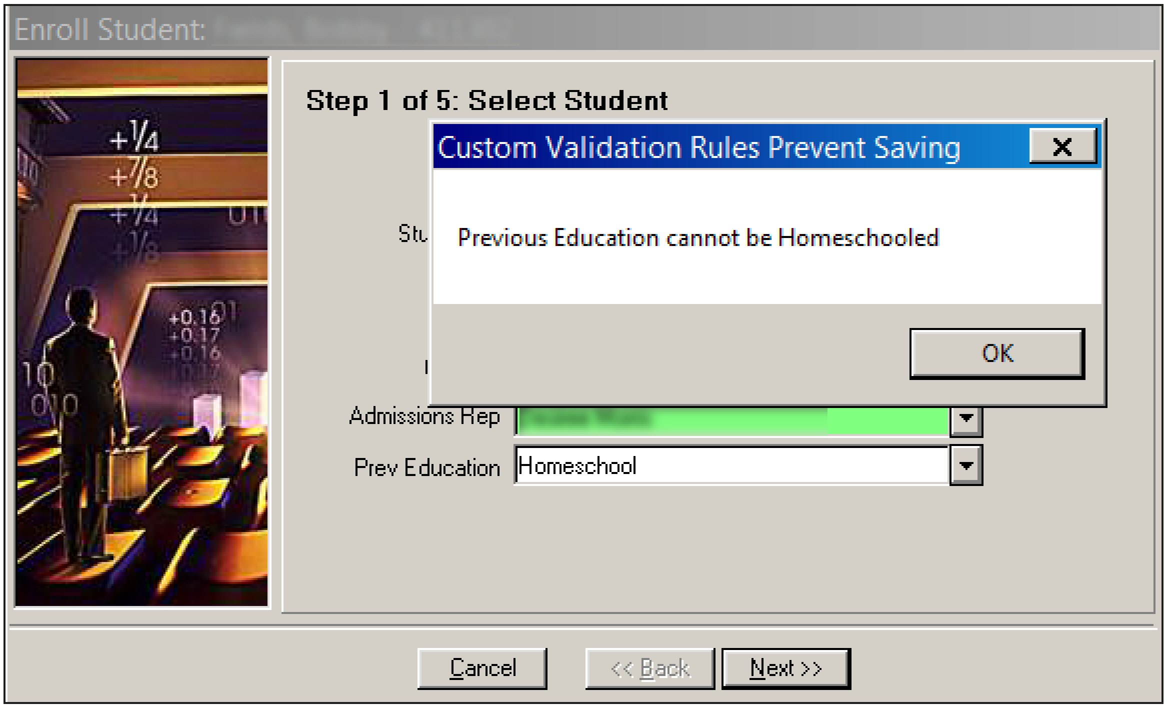
Sequence

LogLine

Text: "STEP 1 ENROLL WIZ: " & Environm

Level: Trace

The custom validation message for this case is as follows:



Enroll Student: [redacted]

Step 1 of 5: Select Student

Custom Validation Rules Prevent Saving

Previous Education cannot be Homeschooled

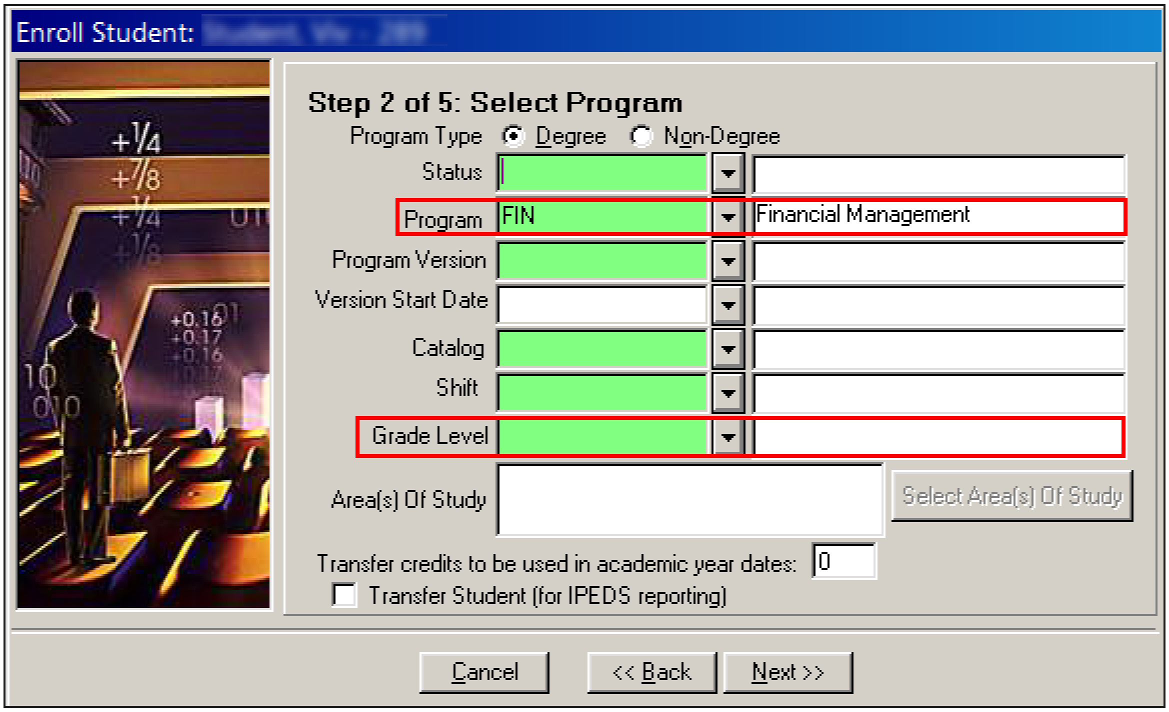
Admissions Rep [redacted]

Prev Education: Homeschool

2. The second case performs custom validations on the **Program** and **Grade Level** selections.

    - If the student's residence is in Alaska, the student is not allowed to enroll in a "Golf" program.
    - If Grade Level High School (8) is selected, an error message states "Grade Level cannot be High School".

    Note: You can hard code the grade level value to compare to (i.e., "8") or use a Look-upReferenceItem activity instead.



Enroll Student: [redacted]

Step 2 of 5: Select Program

Program Type ● Degree ○ Non-Degree

Status

Program: FIN Financial Management

Program Version

Version Start Date

Catalog

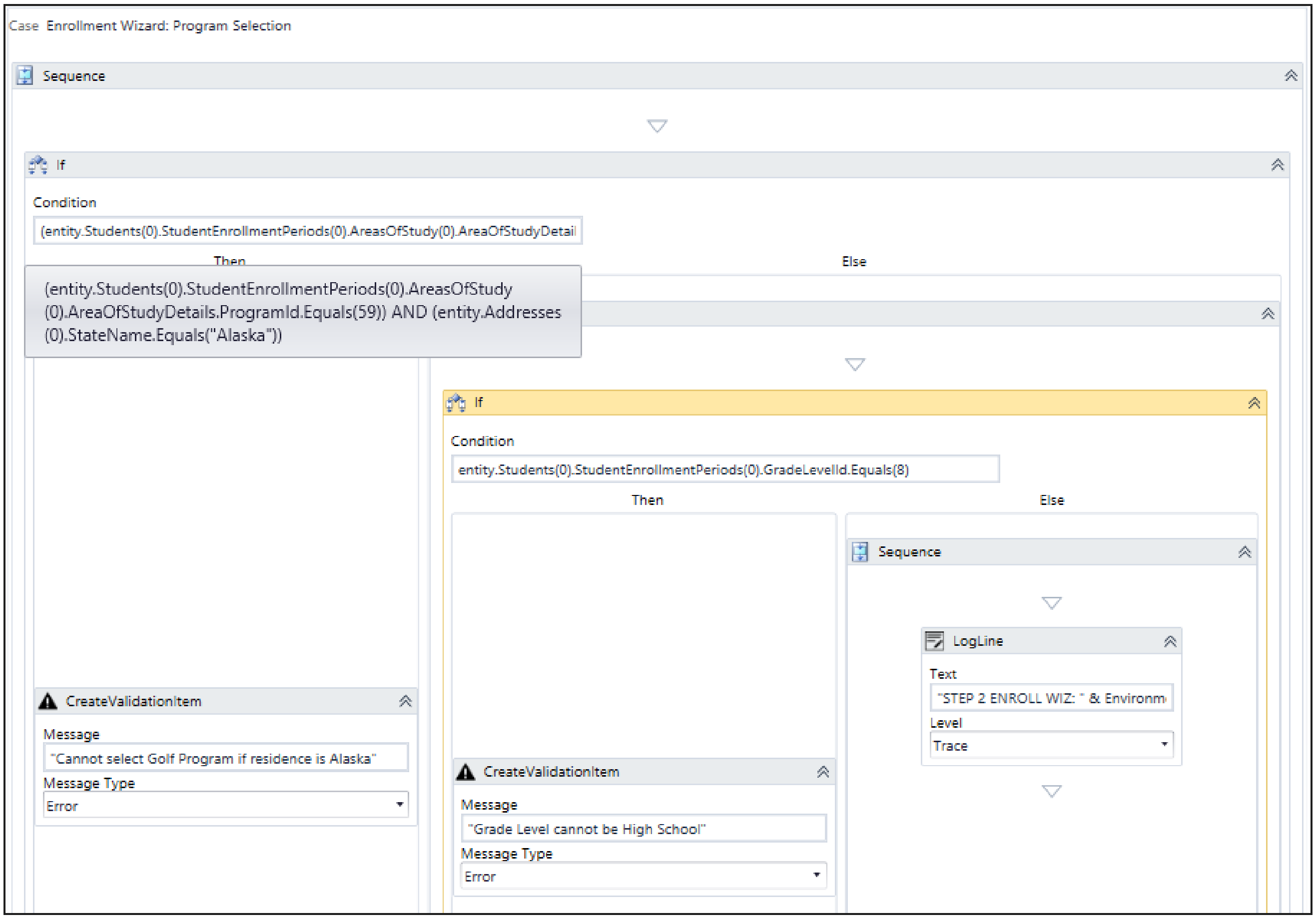
Shift

Grade Level

Area(s) Of Study

Transfer credits to be used in academic year dates: 0

□ Transfer Student (for IPEDS reporting)



Case Enrollment Wizard: Program Selection

Sequence

If

Condition

(entity.Students(0).StudentEnrollmentPeriods(0).AreasOfStudy(0).AreaOfStudyDetail

(entity.Students(0).StudentEnrollmentPeriods(0).AreasOfStudy(0).AreaOfStudyDetails.ProgramId.Equals(59)) AND (entity.Addresses(0).StateName.Equals("Alaska"))

Then

Else

CreateValidationItem

Message

"Cannot select Golf Program if residence is Alaska"

Message Type

Error

If

Condition

entity.Students(0).StudentEnrollmentPeriods(0).GradeLevelId.Equals(8)

Then

Else

CreateValidationItem

Message

"Grade Level cannot be High School"

Message Type

Error

Sequence

LogLine

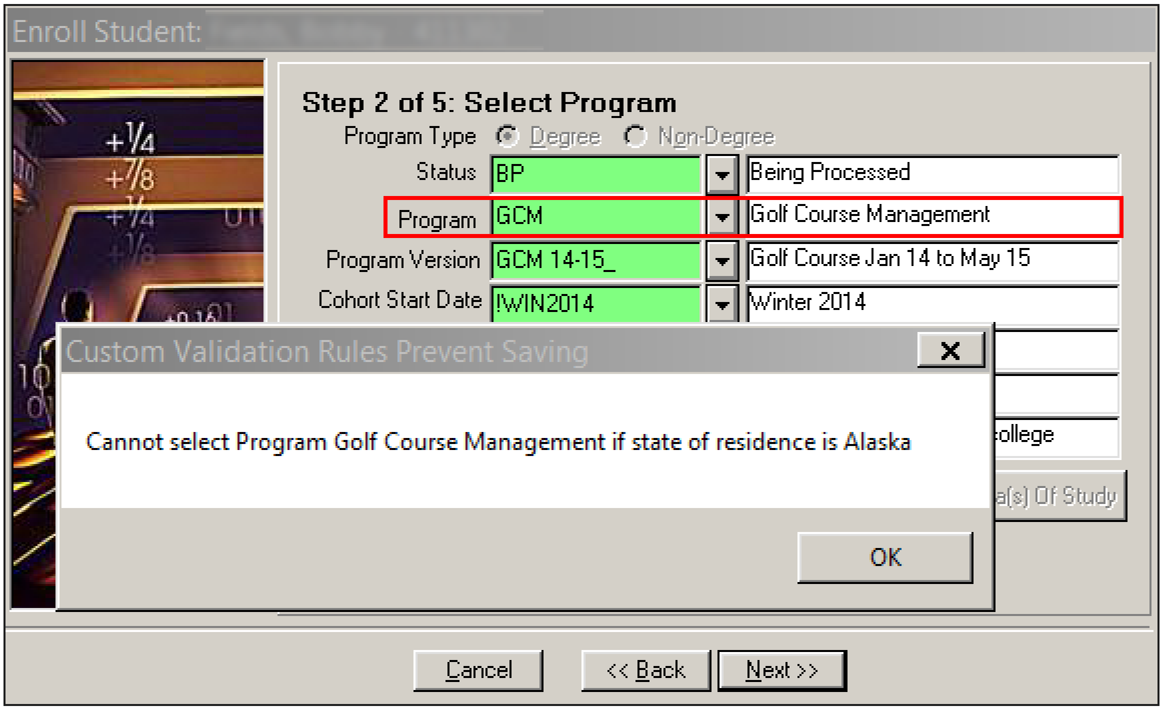
Text

"STEP 2 ENROLL WIZ: " & Environm

Level

Trace

The custom validation message for this case is as follows:



Enroll Student: [redacted]

Step 2 of 5: Select Program

Program Type ● Degree ○ Non-Degree

Status BP Being Processed

Program GCM Golf Course Management

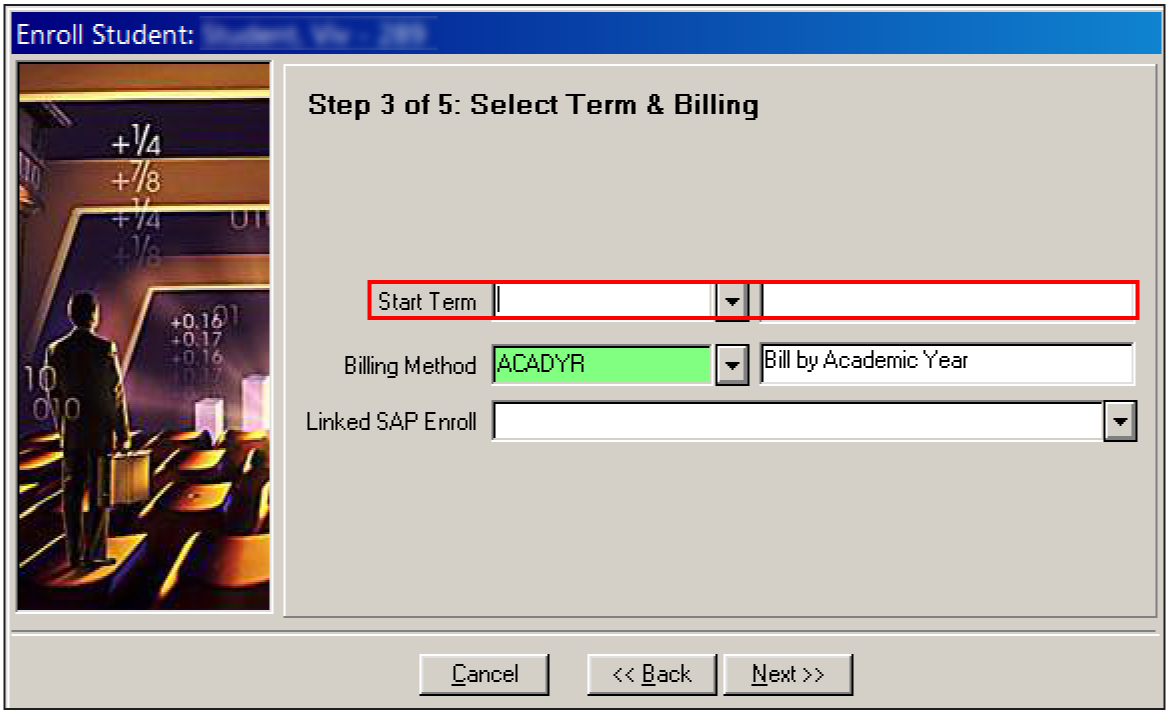
Program Version GCM 14-15_ Golf Course Jan 14 to May 15

Cohort Start Date !WIN2014 Winter 2014

Custom Validation Rules Prevent Saving

Cannot select Program Golf Course Management if state of residence is Alaska

3. The third case checks for an entry in the **Start Term** field.



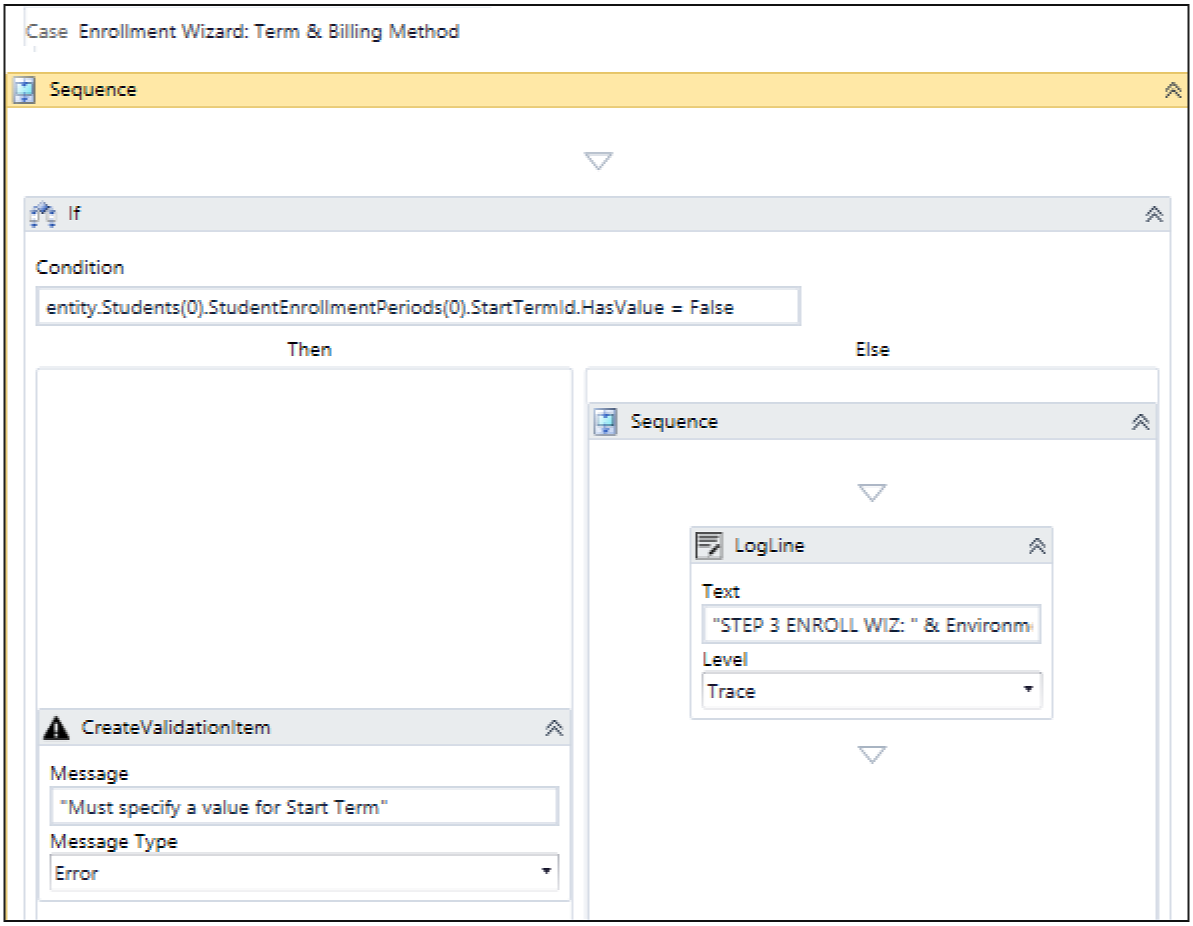
Enroll Student: [redacted]

Step 3 of 5: Select Term & Billing

Start Term:

Billing Method: ACADYR — Bill by Academic Year

Linked SAP Enroll:



Case Enrollment Wizard: Term & Billing Method

Sequence

If

Condition: entity.Students(0).StudentEnrollmentPeriods(0).StartTermId.HasValue = False

Then

CreateValidationItem

Message: "Must specify a value for Start Term"

Message Type: Error

Else

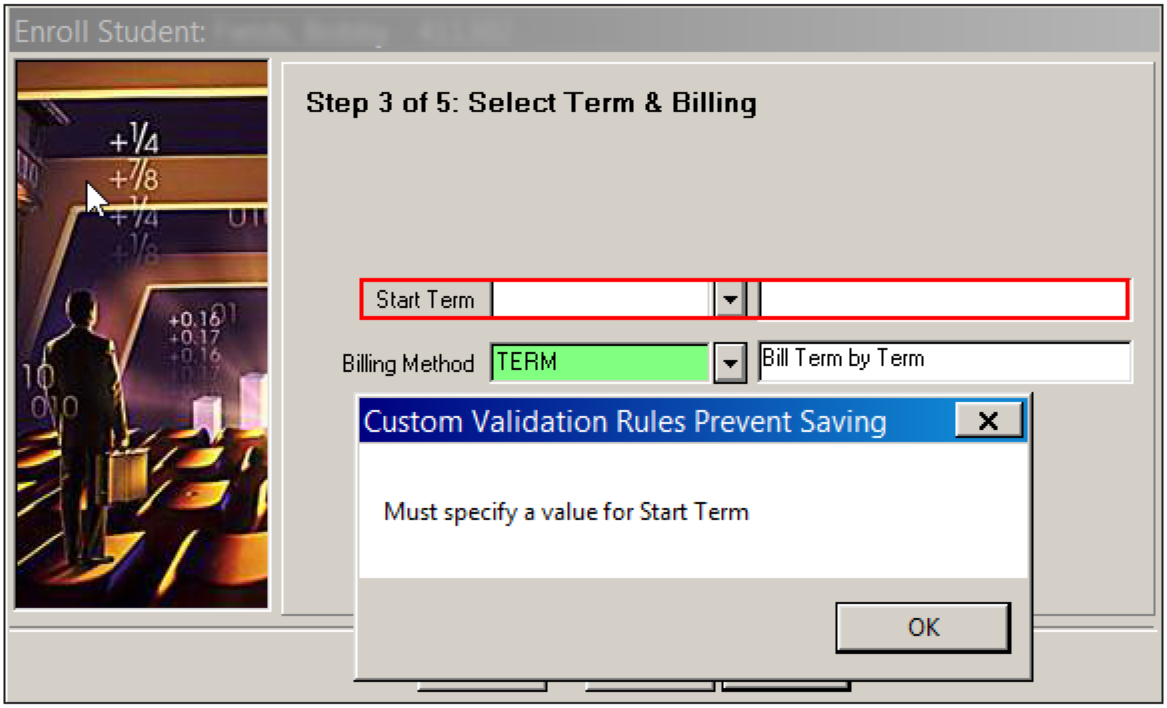
Sequence

LogLine

Text: "STEP 3 ENROLL WIZ: " & Environm

Level: Trace

The custom validation message for this case is as follows:



Enroll Student: [redacted]

Step 3 of 5: Select Term & Billing

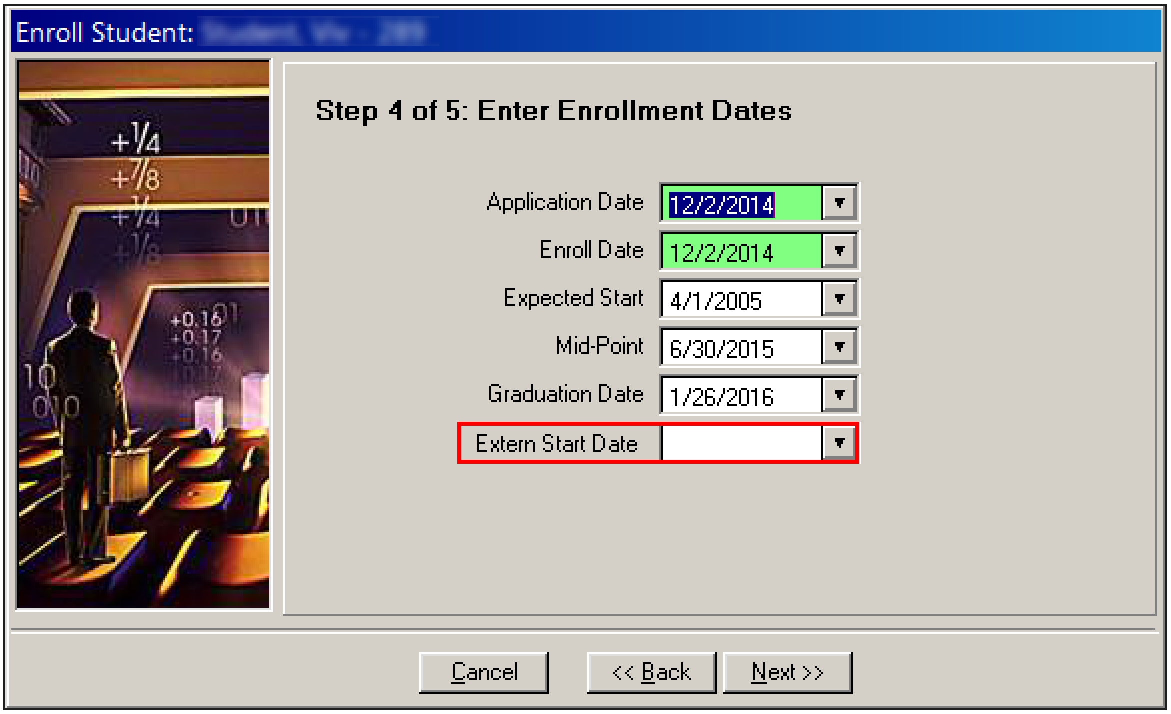
Start Term

Billing Method: TERM — Bill Term by Term

Custom Validation Rules Prevent Saving

Must specify a value for Start Term

4. The fourth case checks for an entry in the **Extern Start Date** field.



Enroll Student: [redacted]

Step 4 of 5: Enter Enrollment Dates

Application Date: 12/2/2014

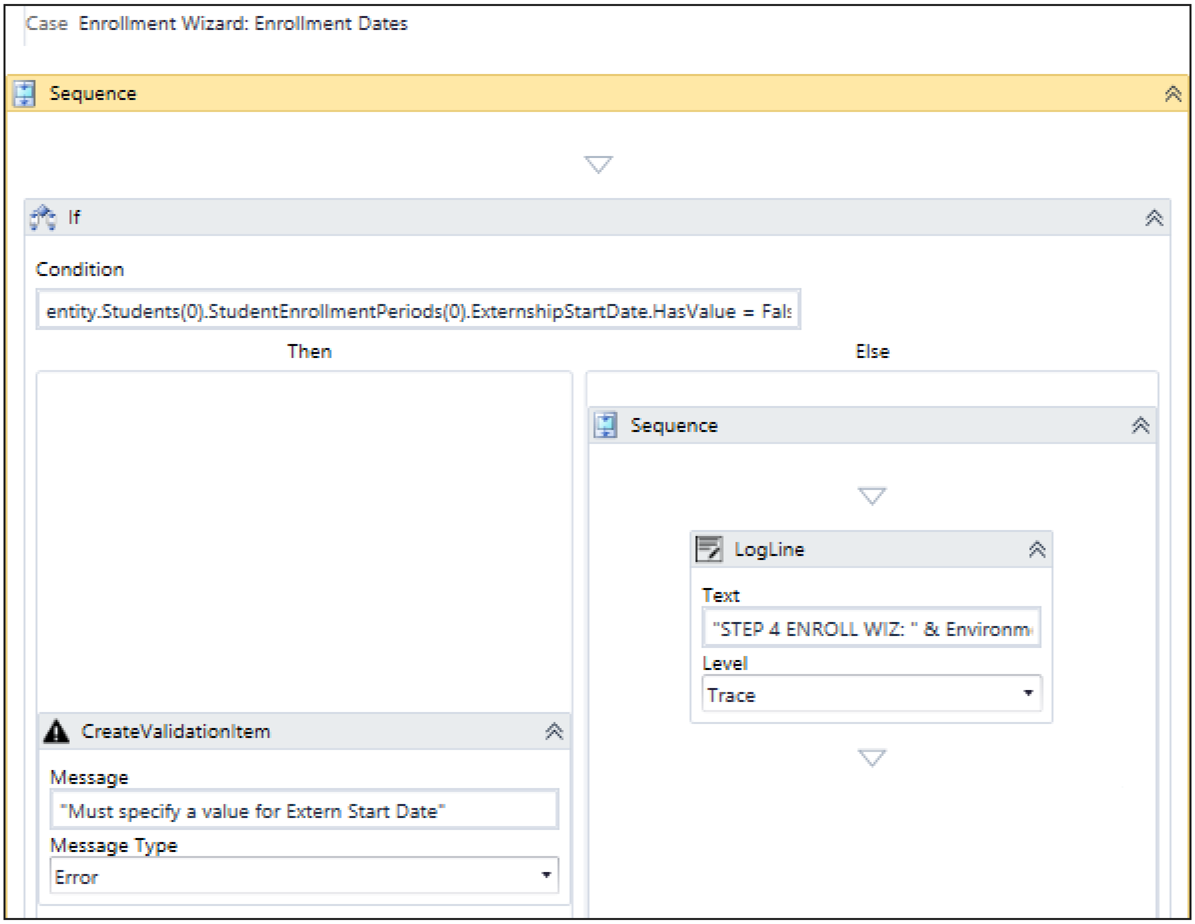
Enroll Date: 12/2/2014

Expected Start: 4/1/2005

Mid-Point: 6/30/2015

Graduation Date: 1/26/2016

Extern Start Date:



Case Enrollment Wizard: Enrollment Dates

Sequence

If

Condition

entity.Students(0).StudentEnrollmentPeriods(0).ExternshipStartDate.HasValue = Fals

LogLine

Text: "STEP 4 ENROLL WIZ: " & Environm

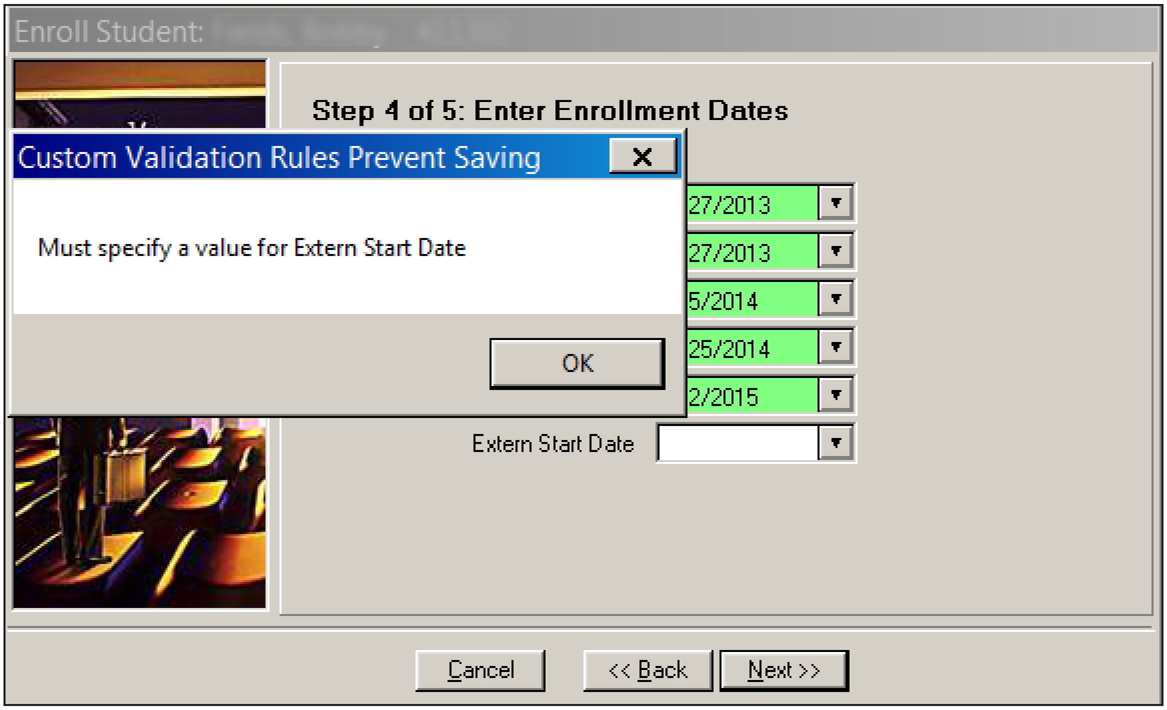
Level: Trace

CreateValidationItem

Message: "Must specify a value for Extern Start Date"

Message Type: Error

The custom validation message for this case is as follows:



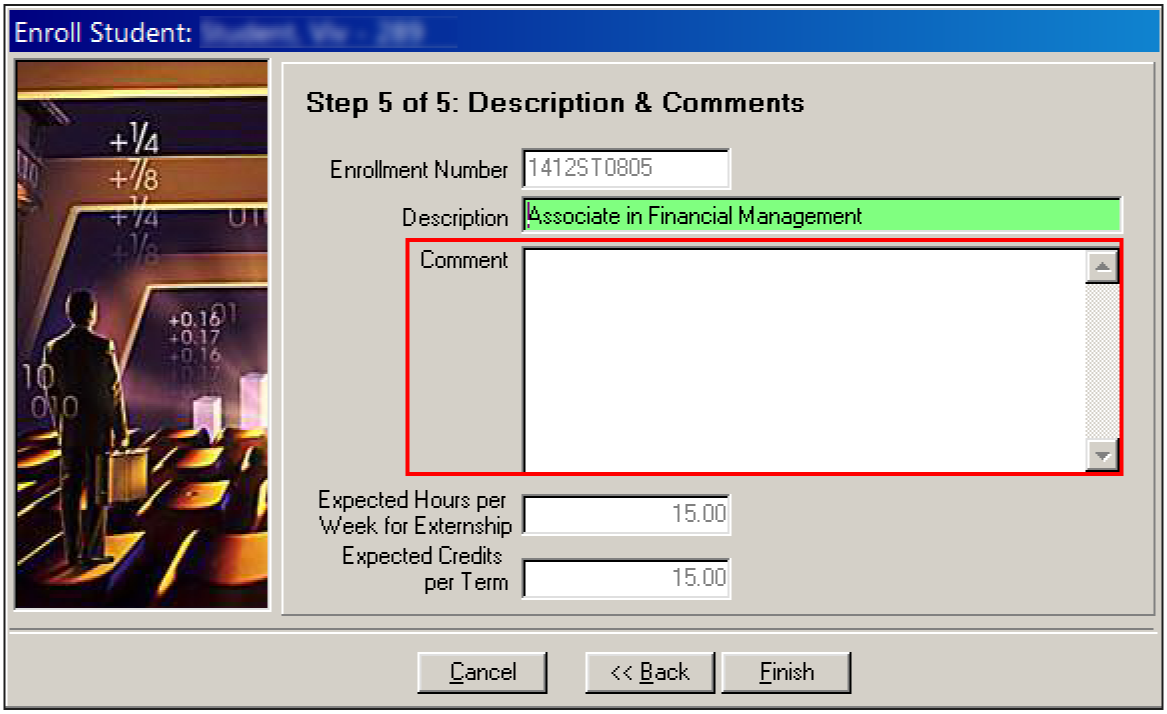
Enroll Student: [redacted]

Step 4 of 5: Enter Enrollment Dates

Custom Validation Rules Prevent Saving

Must specify a value for Extern Start Date

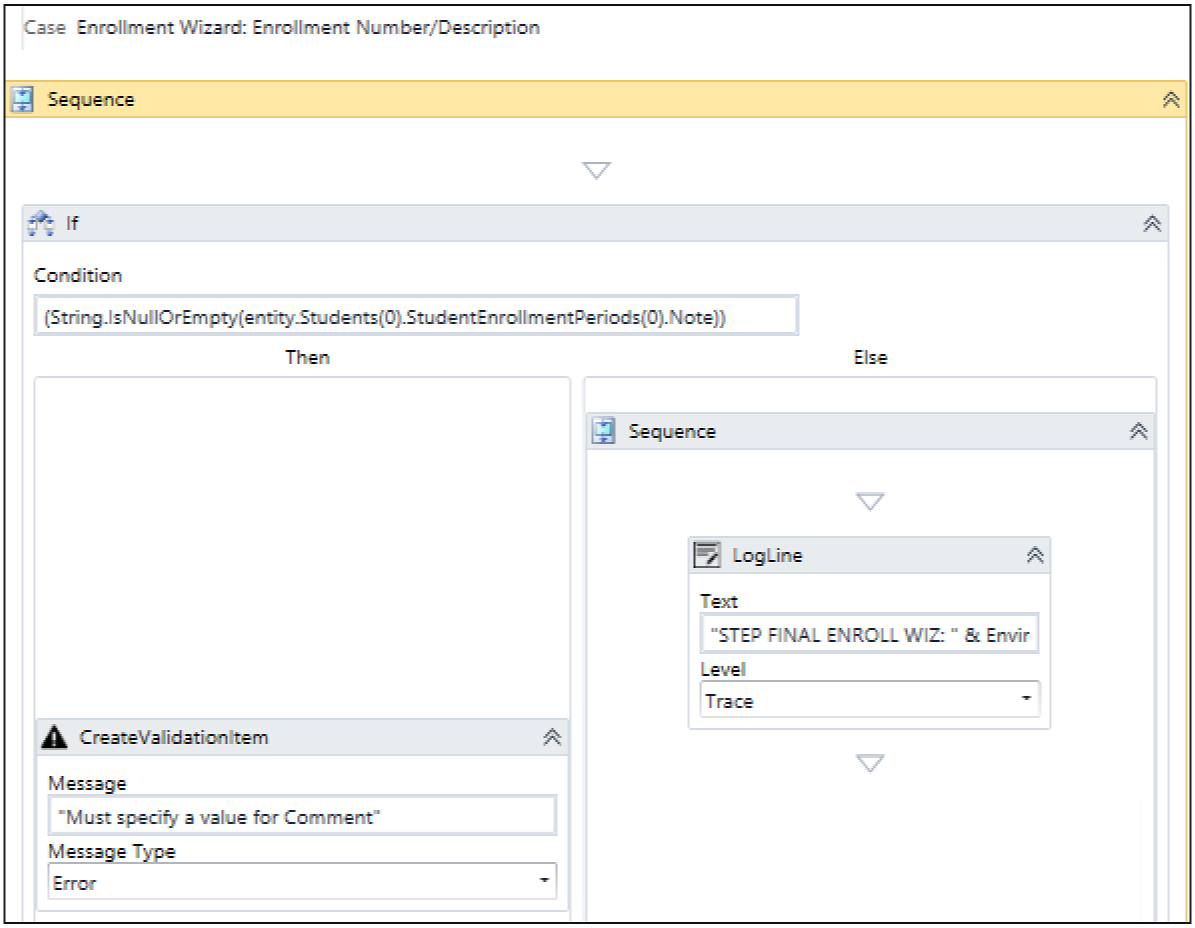
5. The last case checks for an entry in the **Comment** field.



Enroll Student: [redacted]

Step 5 of 5: Description & Comments

Enrollment Number	1412ST0805
Description	Associate in Financial Management
Comment	
Expected Hours per Week for Externship	15.00
Expected Credits per Term	15.00



Case Enrollment Wizard: Enrollment Number/Description

Sequence

If

Condition

(String.IsNullOrEmpty(entity.Students(0).StudentEnrollmentPeriods(0).Note))

Then

CreateValidationItem

Message: "Must specify a value for Comment"

Message Type: Error

Else

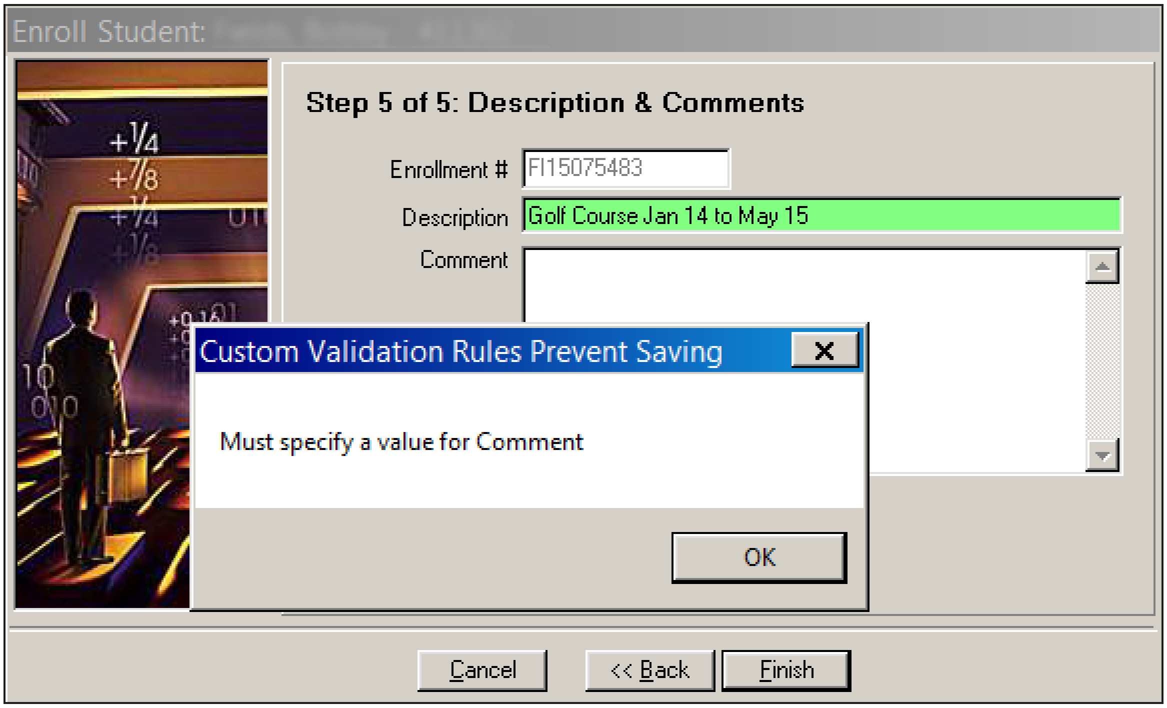
Sequence

LogLine

Text: "STEP FINAL ENROLL WIZ: " & Envir

Level: Trace

The custom validation message for this case is as follows:



Enroll Student: [redacted]

Step 5 of 5: Description & Comments

Enrollment # FI15075483

Description Golf Course Jan 14 to May 15

Comment

Custom Validation Rules Prevent Saving

Must specify a value for Comment

Long Running Workflow

A human workflow, or long running workflow, refers to a type of business process where time elapses between actions, usually waiting for some person to make a decision, which then resumes the workflow. In most cases these workflows refer to approval processes. For example, a student makes a request and that request requires a notification to be sent to one or more approvers.

To create a long running workflow, you will need the following:

- Specify the entity and event that will initialize the process, for example, a document being requested.
- Get the workflow instance.
- Save the workflow instance to a location where it can be retrieved.
- Persist the workflow through a bookmark or time delay.
- Trigger an en event that resumes the workflow.
- Fetch the workflow instance.
- Complete the workflow or repeat the persist / resume process.

When designing approval processes, you can have a scenario where a single event can continue the workflow or several events need to occur to continue the workflow.

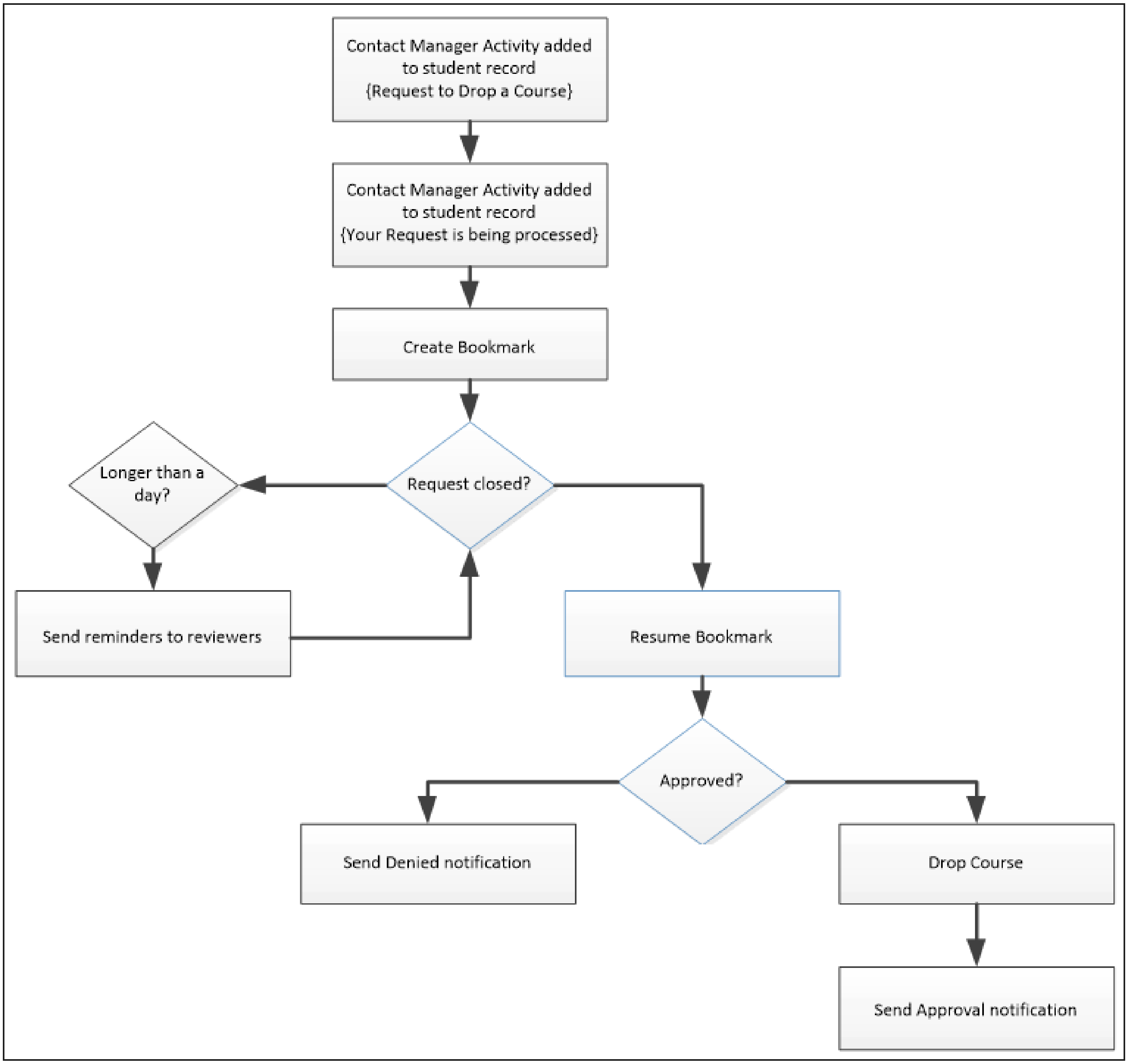
- For singular approval, like approved or denied, use the **Pick** and **Pick Branch** activities to resume bookmarks.
- For multiple approvals, like approver, document, and fee, use a **Parallel** activity to resume bookmarks.

Currently, you can only save the WorkflowInstanceId to the CmDocument or CmEvent tables in Anthology Student. Most approvals or long running workflows will be related to a Contact Manager activity or a document.

Scenario: Request Approval from a User

We will add a Contact Manager activity to a student and assign it to an advisor. The first workflow will “wait” until the advisor approves. The second workflow will "wake up" the first workflow when the Contact Manager activity is closed.

The following flowchart illustrates the business process that is captured in the workflow sequence.


Contact Manager Activity added to student record {Request to Drop a Course}
Contact Manager Activity added to student record {Your Request is being processed}
Create Bookmark
Request closed?
Longer than a day?
Send reminders to reviewers
Resume Bookmark
Approved?
Send Denied notification
Drop Course
Send Approval notification

Prerequisites

- A Contact Manager activity is set up in Anthology Student.
- Access to the workflow logs is available.

Workflow Activities Used

The following activities will be used in the workflow:

- If
- Sequence
- LookupReferenceItem
- CreateBookmark
- ResumeBookmark
- Pick
- PickBranch
- LogLine
- ExecuteNonQuery
- ExecuteDataReader
- GetWorkflowInstanceId
- FlowChart

Continue with Create a Long Running Workflow.

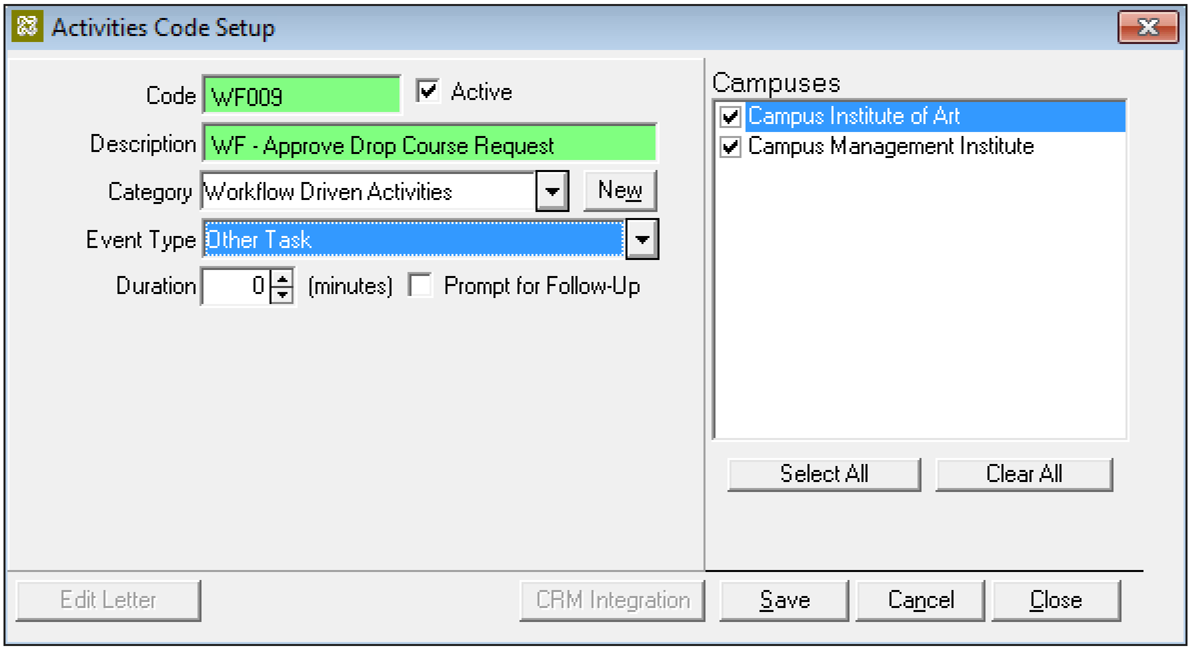
Create a Long Running Workflow

When the "Approve Drop Request" Contact Manager activity is added to a student’s record, this workflow detects the event and waits for the activity to be closed before executing the logic in the Pick activity.

Step 1: Create a Contact Manager activity in Anthology Student

a. In Anthology Student, navigate to **Lists** > **Contact Manager** > **Activities** > **New**.

b. Add a new Contact Manager activity for the workflow. In this example, we created the **Other Task** activity with the name **WF – Approve Drop Course Request**.



Activities Code Setup

Code WF009 ☑ Active

Description WF - Approve Drop Course Request

Category Workflow Driven Activities

Event Type Other Task

Duration 0 (minutes) □ Prompt for Follow-Up

Campuses

☑ Campus Institute of Art

☑ Campus Management Institute

c. Navigate to **Lists** > **Contact Manager** > **Activity Result** > **New**.

d. Create the following Contact Manager activity results:

  - **Approve Drop Course Request**
  - **Deny Drop Course Request**

  Make sure these activity results are assigned to **Other Task** types.

Step 2: Create a workflow

a. Launch Workflow Designer.

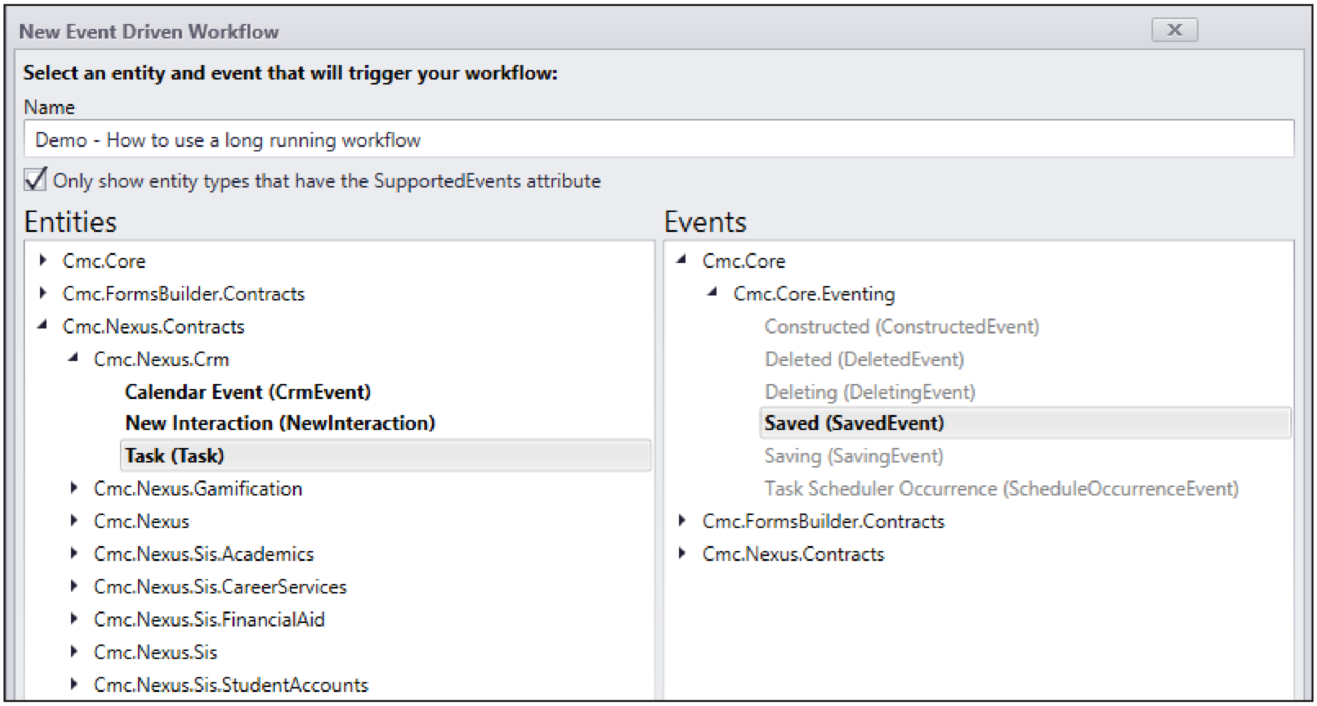
b. On the Home tab, click **New Event Workflow**.

c. In the *Name* field, type **Demo - How to use a long running workflow**.

d. In the Entities area, expand Cmc.Nexus.Contracts > Cmc.Nexus.Crm and select **Task {Task)**.

e. In the Events area, expand Cmc.Core.Eventing and select **Saved (SavedEvent)**.

f. Click **OK**.



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name: Demo - How to use a long running workflow

☑ Only show entity types that have the SupportedEvents attribute

Entities

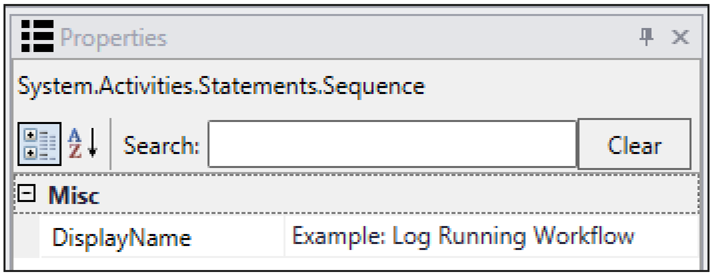
- Cmc.Core
- Cmc.FormsBuilder.Contracts
- Cmc.Nexus.Contracts
  - Cmc.Nexus.Crm
    - **Calendar Event (CrmEvent)**
    - **New Interaction (NewInteraction)**
    - **Task (Task)**
  - Cmc.Nexus.Gamification
  - Cmc.Nexus
  - Cmc.Nexus.Sis.Academics
  - Cmc.Nexus.Sis.CareerServices
  - Cmc.Nexus.Sis.FinancialAid
  - Cmc.Nexus.Sis
  - Cmc.Nexus.Sis.StudentAccounts

Events

- Cmc.Core
  - Cmc.Core.Eventing
    - Constructed (ConstructedEvent)
    - Deleted (DeletedEvent)
    - Deleting (DeletingEvent)
    - **Saved (SavedEvent)**
    - Saving (SavingEvent)
    - Task Scheduler Occurrence (ScheduleOccurrenceEvent)
- Cmc.FormsBuilder.Contracts
- Cmc.Nexus.Contracts

Step 3: Rename the default sequence

In the Properties pane, set the DisplayName to **Example: Log Running Workflow**.



System.Activities.Statements.Sequence

DisplayName: Example: Log Running Workflow

Step 4: Create variables

a. In the Variables pane, create a variable named **DropActivity**. This variable will store the Contact Manager activity we created in Anthology Student.

b. In the Variable type field, click **Browse for Types** and navigate to **Cmc.Nexus.Common.Workflow > LookupReferenceItem**. This type is required because we are going to look up the Contact Manager activity using a workflow activity

c. Create a second variable named **WorkflowId**. This variable will store the workflow instance Id.

d. In the Variable type field, click **Browse for Types** and navigate to **mscorlib > System > Guid**.

Step 5: Look up the Contact Manager activity

a. In the Toolbox under CMC.Nexus.Workflow, find the **LookupReferenceItem** activity and drag it into the sequence.

b. In the Properties pane, change the DisplayName to **Find Activity**.

c. In the Reference Item Type field, select **Task Template**.

d. In the Reference Item field, select **WF – Approve Drop Course Request**. This is the Contact Manager activity you created earlier.

e. In the Properties pane, set the Reference Item to the **DropActivity** variable.

Step 6: Write to the log

a. In the Toolbox under CMC.Core.Workflow.Activities, find the **LogLine** activity and drag it into the sequence under the LookupReferenceItem activity.

   The LogLine activity writes to the log file as the workflow goes a long. It is a great way to see what is happening with the workflow and helps during troubleshooting while learning or building your workflow. It is similar to commenting your code.

b. In the Properties pane, change the DisplayName to **Initialize**.

c. In the Text field, write any text that you want to show up in the log, for example,

"Starting Long Running Workflow Example - The Activity we are looking for is " + DropActivity.Name + " with the TaskTemplateId = " + entity.TaskTypeId.ToString

d. Leave the Level set to **Information**. Depending on how your NLog is set up, different levels are logged in different ways.

Step 7: Make sure this is the activity we are looking for

a. In the Toolbox under Control Flow, find the **If** activity and drag it into the sequence under the LogLine activity.

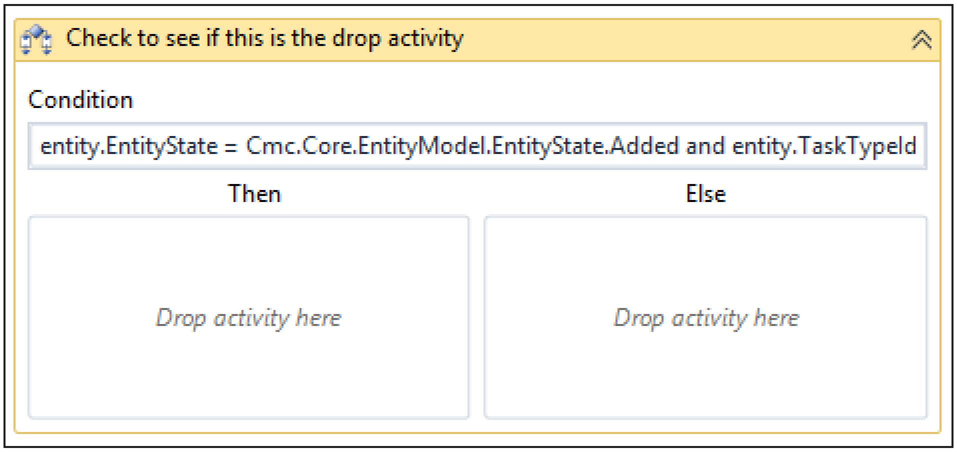
   The If condition will check if the event that occurred is the one we are looking for. We are looking for the DropActivity event (see step 5).

b. In the Properties pane, change the DisplayName to **Check to see if this is the drop activity**.

c. In the Condition field, type the following:

   entity.EntityState = Cmc.Core.EntityModel.EntityState.Added and entity.TaskTypeId = DropActivity.Id

   When a Contact Manager activity is added, this condition checks if the activity is a drop activity; if it is, the workflow continues, else the workflow ends.



Check to see if this is the drop activity

Condition

entity.EntityState = Cmc.Core.EntityModel.EntityState.Added and entity.TaskTypeId

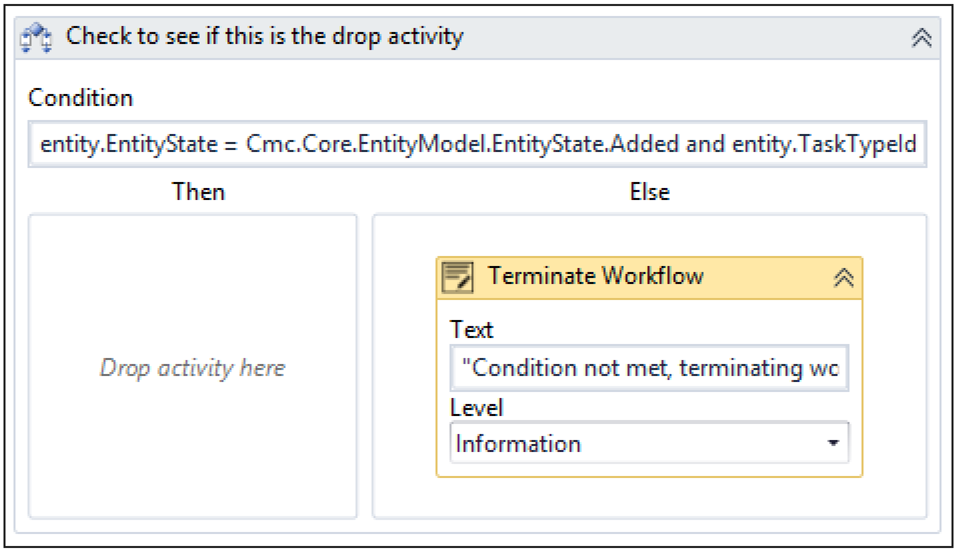
Step 8: Write to the log

a. In the Toolbox under CMC.Core.Workflow.Activities find the **LogLine** activity and drag it into the Else block of the If activity.

b. In the Properties pane, change the DisplayName to **Terminate Workflow**.

c. In the Text field, type **"Condition not met, terminating workflow"**

d. Leave the Level set to **Information**.



Check to see if this is the drop activity

Condition

entity.EntityState = Cmc.Core.EntityModel.EntityState.Added and entity.TaskTypeId

Terminate Workflow

Text

"Condition not met, terminating wc

Level

Information

Step 9: Get the WorkflowInstanceId

a. In the Toolbox under Control Flow, find the **Sequence** activity and drag it into the Then block of the If activity.

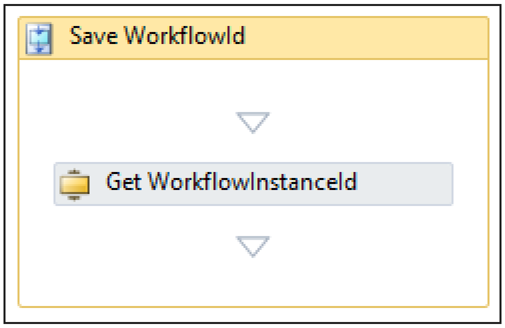
   You can only have one activity inside the Then and Else blocks of the If activity. But a Sequence is an activity that allows you to have multiple workflow activities.

b. In the Properties pane, change the DisplayName to **Save WorkflowId**.

c. In the Toolbox under CMC.Core.Workflow.Activities, find the **GetWorkflowInstanceId** activity and drag it into the sequence.

d. In the Properties pane, set the Result field to the variable **WorkflowId**.

e. Optional: In the DisplayName field, add a space between Get and WorkflowInstanceId to make it easier to read.



Save WorkflowId

Get WorkflowInstanceId

Step 10: Save the WorkflowInstanceId

We are going to save the WorkflowInstanceId to the CmEvent record in the Anthology Student database so that we can recall this workflow later. Since we are working with Anthology Student, we will not need a connection to the database. We just need to update the WorkflowInstanceID column that was added as part of Anthology Student 16.1.0.

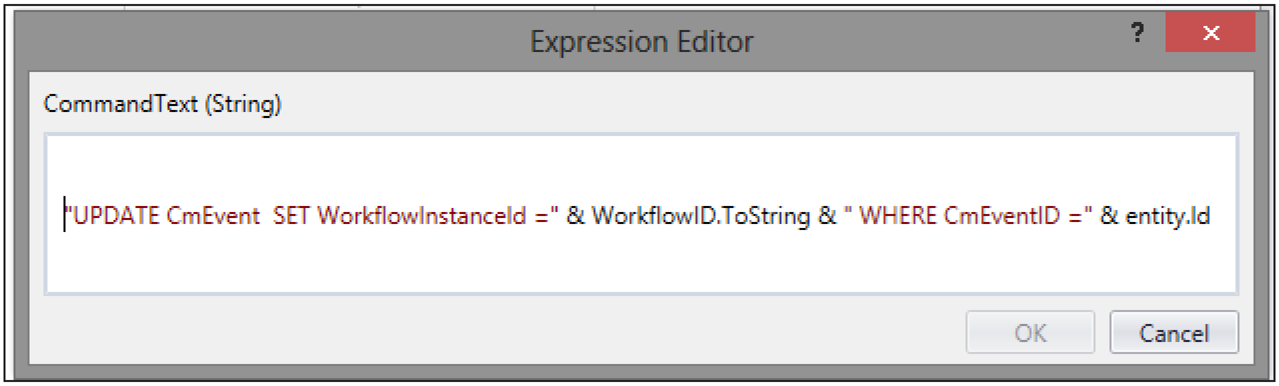
a. In the Toolbox under Control Flow, find the **ExecuteNonQuery** activity and drag it into the sequence under the GetWorkflowInstanceId activity.

b. In the Properties pane, change the DisplayName to **Save the Workflow Instance**.

c. In the Command field, type the following:

   "UPDATE CmEvent SET WorkflowInstanceId =" & WorkflowID.ToString & " WHERE CmEventID =" & entity.Id

   Note: SQL commands need to be strings, that is, quotes are required.



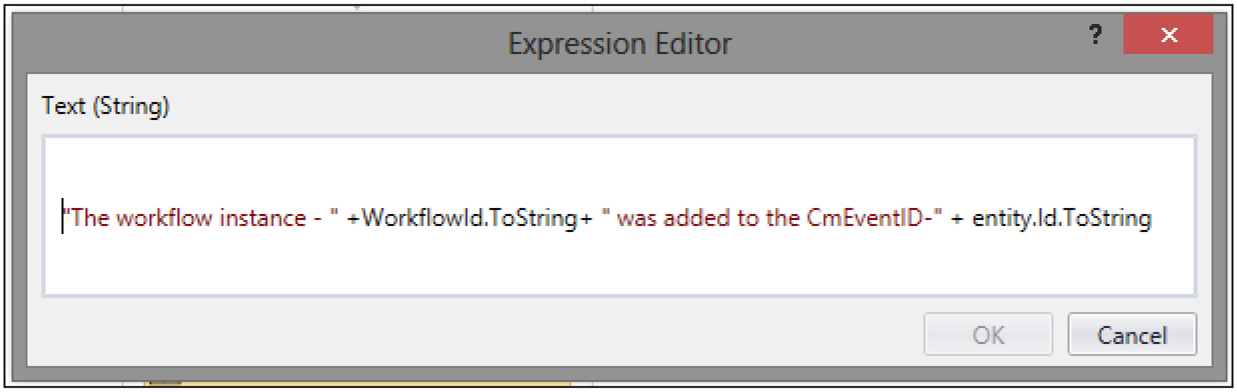
```
"UPDATE CmEvent  SET WorkflowInstanceId =" & WorkflowID.ToString & " WHERE CmEventID =" & entity.Id
```

d. In the Toolbox under CMC.Core.Workflow.Activities, find the **LogLine** activity and drag it under the ExecuteNonQuery activity.

e. In the Properties pane, change the DisplayName to **Update Log**.

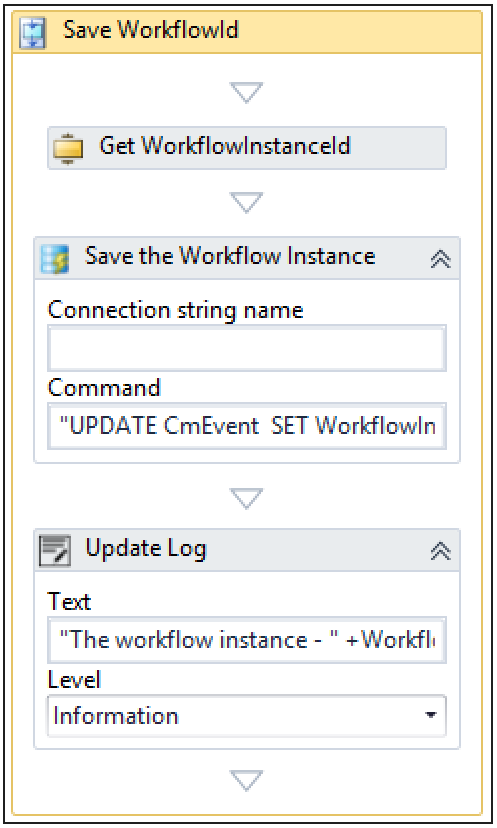
f. In the Text field, type the following:

   “The workflow instance - " +WorkflowId.ToString+ " was added to the CmEventID-" + entity.Id.ToString”



```
"The workflow instance - " +WorkflowId.ToString+ " was added to the CmEventID-" + entity.Id.ToString
```

g. Set the Level to **Information**.



Connection string name

Command
"UPDATE CmEvent SET WorkflowIn

Text
"The workflow instance - " +Workfl

Level
Information

Step 11: Pause the workflow

Because we only have two conditions, approved or denied, we will use the Pick activity to pick which business process will resume once the approver closes the Contact Manager activity with an approved or denied result.

a. In the Toolbox under Control Flow, find the **Pick** activity and drag it under the If activity.

   The Pick activity uses PickBranch activities, one for each branch of the business process that will execute when the workflow resumes.

b. In the Properties pane, change the DisplayName to **Approved or Denied Process**.

c. In the Toolbox under Control Flow, find the **PickBranch** activity and drag it into the Pick activity.

d. In the Properties pane, change the DisplayName to **Approved Process**.

e. In the Toolbox under Cmc.Core.Workflow.Activities, find the **CreateBookmark** activity and drag it into the Trigger section of the PickBranch.

   The CreateBookmark activity is saved and referenced later based on the approvers actions.

f. In the Properties pane, change the DisplayName to **Pause the workflow**.

g. Set the BookmarkName property to **“Approved”**.

h. Drag a **LogLine** activity into the Action section of the PickBranch.

i. In the Text field, type **“The Request was Approved”**.

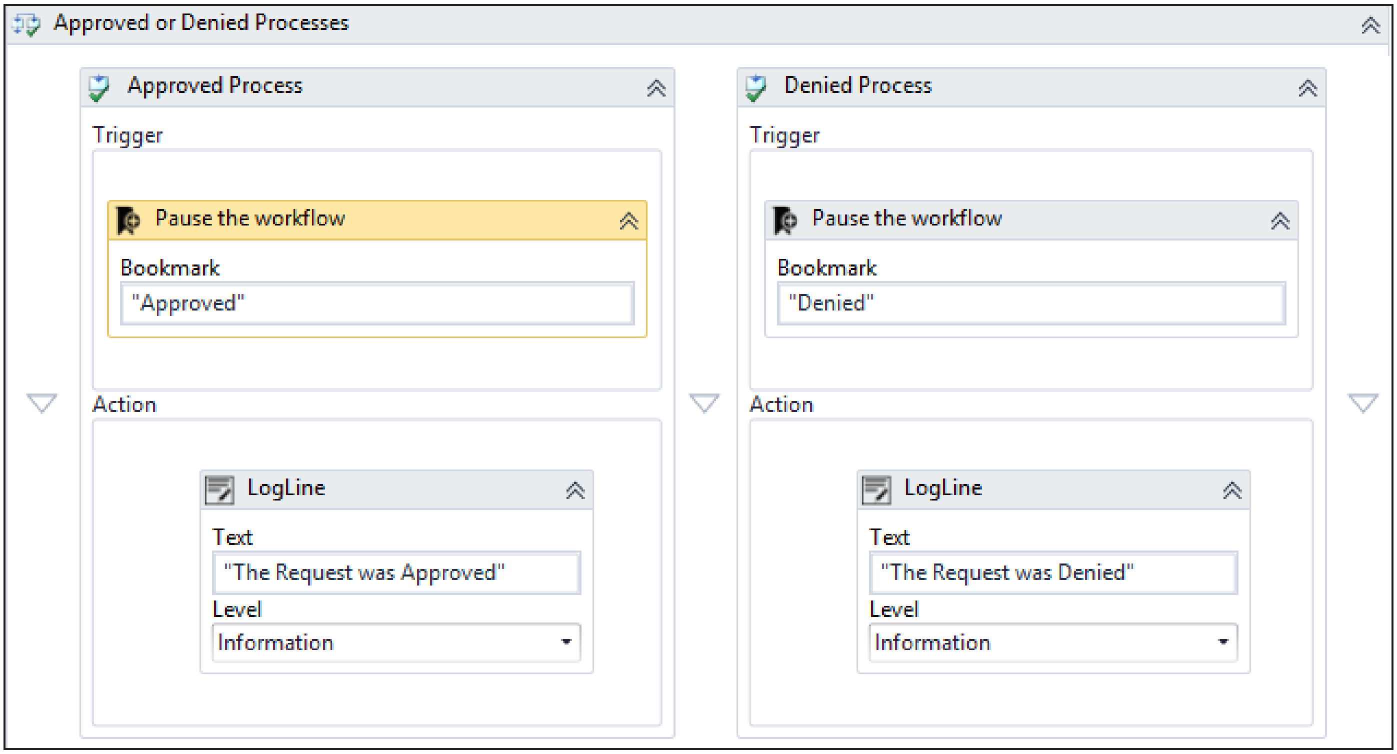
j. Right-click the PickBranch activity and select **Copy** .

k. Right-click next to the PickBranch activity and select **Paste**. We now have two pick branches.

l. In the Properties pane of the copied PickBranch, change the DisplayName to **Denied Process**.

m. Drag a **LogLine** activity into the Action section of the copied PickBranch.

n. In the Text field, type **“The Request was Denied”**.



Approved or Denied Processes

Approved Process

Trigger

Pause the workflow

Bookmark "Approved"

Action

LogLine

Text "The Request was Approved"

Level Information

Denied Process

Trigger

Pause the workflow

Bookmark "Denied"

Action

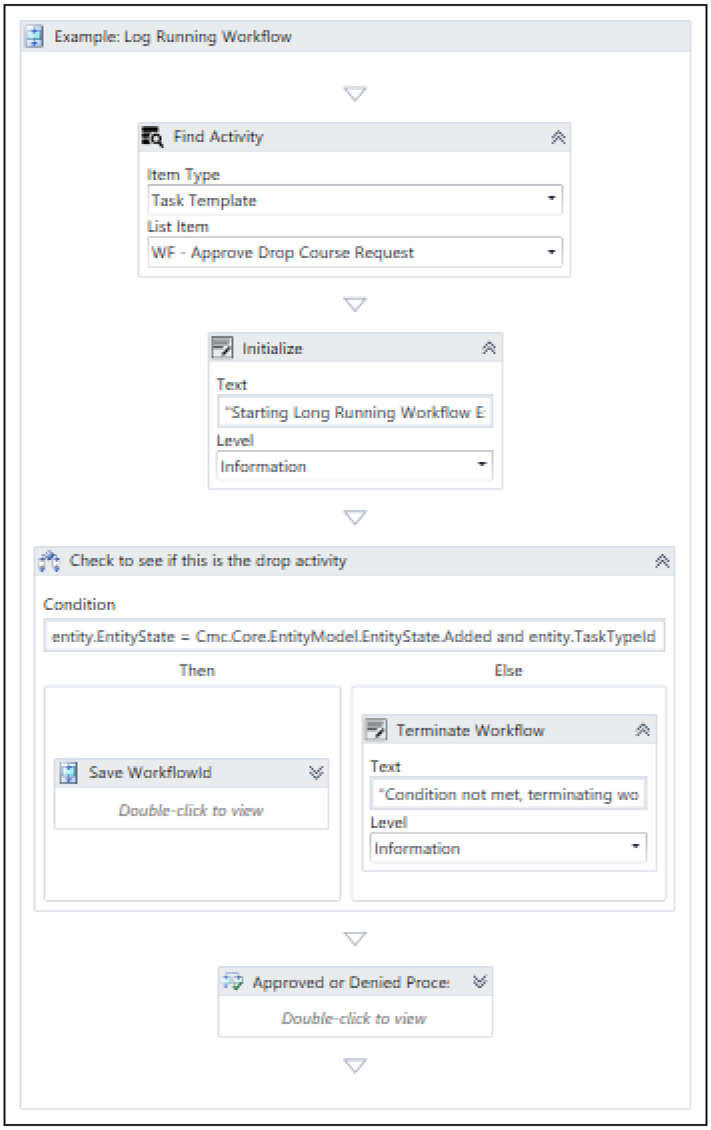
LogLine

Text "The Request was Denied"

Level Information

Step 12: Save and publish the workflow

a. Check your workflow. Scroll through the workflow or use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.



Example: Log Running Workflow

Find Activity

Item Type: Task Template

List Item: WF - Approve Drop Course Request

Initialize

Text: "Starting Long Running Workflow E

Level: Information

Check to see if this is the drop activity

Condition: entity.EntityState = Cmc.Core.EntityModel.EntityState.Added and entity.TaskTypeId

Then

Save WorkflowId

Double-click to view

Else

Terminate Workflow

Text: "Condition not met, terminating wo

Level: Information

Approved or Denied Proce:

Double-click to view

b. Click **Publish**. The New Workflow Definition Version window is displayed.

c. Select **Enable This Workflow Version**

d. Click **Publish**, then **Cancel** to close the publisher window.

We now need to create another workflow that will resume this workflow when the Contact Manager activity is closed.

Continue with Wake up the Long Running Workflow.

Wake up the Long Running Workflow

This workflow resumes the long running workflow when an advisor approves or denies a student's request to drop a course and the Contact Manager activity is closed.

Step 1: Create a new workflow

a. Launch Workflow Designer.

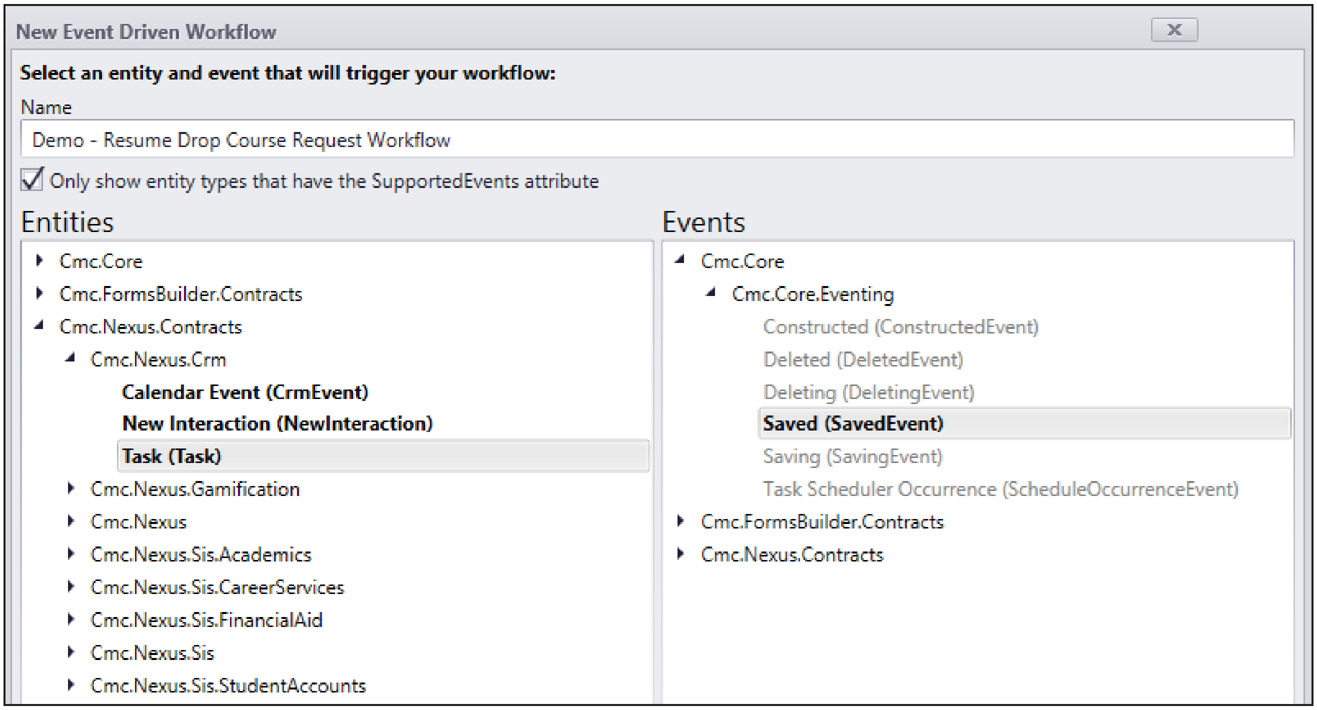
b. On the Home tab, click **New Event Workflow**.

c. In the *Name* field, type **Resume Drop Course Workflow**.

d. In the Entities area, expand Cmc.Nexus.Contracts > Cmc.Nexus.Crm and select **Task {Task)**.

e. In the Events area, expand Cmc.Core.Eventing and select **Saved (SavedEvent)**.

f. Click **OK**.



New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name: Demo - Resume Drop Course Request Workflow

☑ Only show entity types that have the SupportedEvents attribute

Entities
Cmc.Core
Cmc.FormsBuilder.Contracts
Cmc.Nexus.Contracts
Cmc.Nexus.Crm
Calendar Event (CrmEvent)
New Interaction (NewInteraction)
Task (Task)
Cmc.Nexus.Gamification
Cmc.Nexus
Cmc.Nexus.Sis.Academics
Cmc.Nexus.Sis.CareerServices
Cmc.Nexus.Sis.FinancialAid
Cmc.Nexus.Sis
Cmc.Nexus.Sis.StudentAccounts
Events
Cmc.Core
Cmc.Core.Eventing
Constructed (ConstructedEvent)
Deleted (DeletedEvent)
Deleting (DeletingEvent)
Saved (SavedEvent)
Saving (SavingEvent)
Task Scheduler Occurrence (ScheduleOccurrenceEvent)
Cmc.FormsBuilder.Contracts
Cmc.Nexus.Contracts

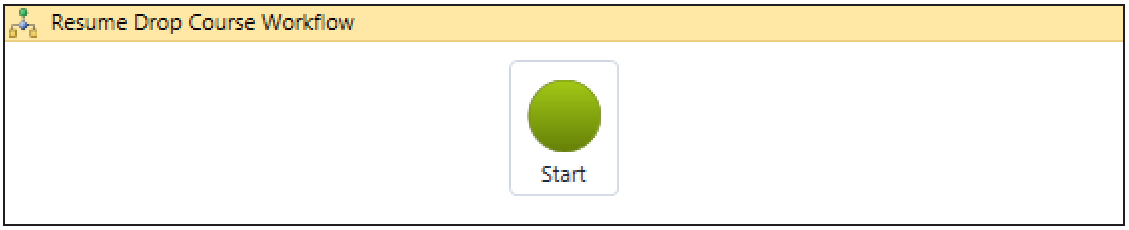
Step 2: Delete the default Sequence

This workflow uses a Flowchart instead of a Sequence. Flowcharts are better used when many decisions need to be considered. In this case, we are really looking for the result of the activity; however, the result is not required in the database. We need to handle approved, denied, and nothing, or a NULL results. This is best done with flow decisions.

a. In the Designer pane, right-click the default Sequence and select **Delete**.

b. In the Toolbox under Flowchart, find the **Flowchart** activity and drag it on to the Designer pane.

c. In the Properties pane, change the DisplayName to **Resume Drop Course Workflow**.



Resume Drop Course Workflow

Step 3: Create variables

In the Variables pane, create variables with the following names and types:

Name	Variable type (see Note)	Default
ActivityResultDenied	LookupReferenceItem (Cmc.Nexus.Common.Workflow)	
ActivityResultApproved		
ActivityStatusClosed		
DropActivity		
BookmarkName	String	"Denied"
Result	Int32	
WorkflowId	Guid (mscorlib > System > Guid)	
Note: If you don’t see the Variable type you need, click **Browse for Types**.		

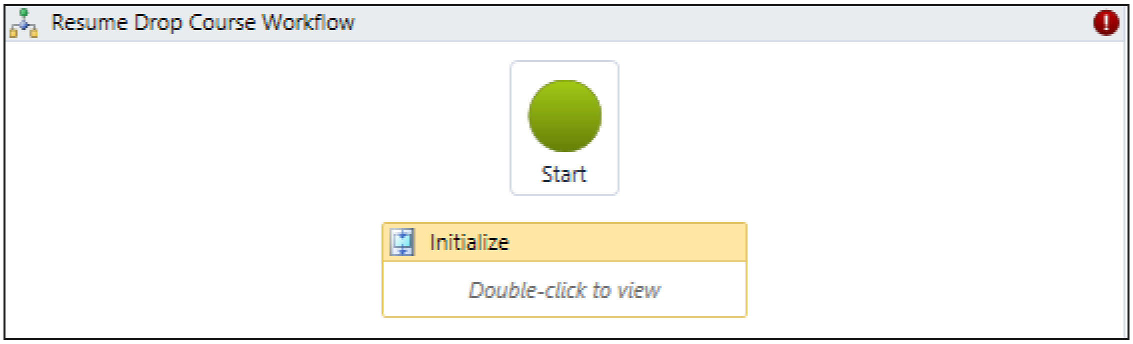
Step 4: Add the Initialize Sequence

Working with flowcharts is nice because you can really organize the steps and activities.

a. Drag a **Sequence** activity below the Start circle of the Flowchart.

b. In the Properties pane, change the DisplayName to **Initialize**.

c. Double-click the **Sequence**. Inside this sequence, we will add some logging and look up some information that we will use in our first flow decision.



Step 5: Write to the log and look up information

a. Drag a **LogLine** activity into the sequence.

b. Drag two **LookupReferenceItem** activities into the sequence.

c. In the LogLine activity, change the DisplayName to **Begin Workflow**.

d. In the Text field of the LogLine activity, type:

"Check event for conditions - the CmEventId is -" + entity.Id.ToString + " the StatusId is - " + entity.TaskStatusId.ToString + " the ResultId is - " + entity.TaskResultId.ToString

e. In the first LookupReferenceItem, specify the following:

DisplayName:	**Find the Drop Activity Status**
Reference Item Type:	**Task Status**
Reference Item:	**ActivityStatusClosed** (This is one of the variables created above.)

f. In the second LookupReferenceItem, specify the following:

DisplayName:	**Find Drop Activity**
Reference Item Type:	**Task Template**
Reference Item:	**DropActivity** (This is one of the variables created above.)

Step 6: Flow Decision

a. Click **Flowchart** in the breadcrumbs at the top of the Designer pane.

b. Drag the **Flow Decision** activity below the sequence we just created.

c. In the Properties pane of the Flow Decision, change the DisplayName to **Drop Activity Closed**.

d. In the Condition field, type:

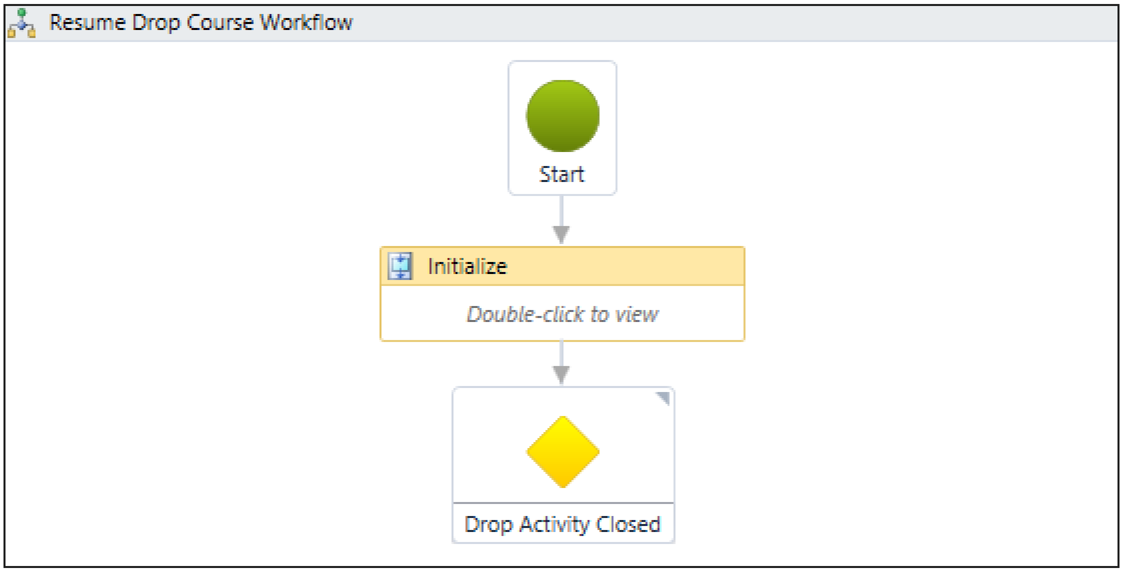
entity.HasChanged(task.TaskStatusIdProperty) and entity.TaskStatusId = ActivityStatusClosed.Id and entity.TaskTypeId = DropActivity.Id

This condition checks if the Contact Manager Drop activity was closed.

- If it isn't the Drop Activity, and the status wasn't what was updated, we will end the workflow.
- If it is the one we are looking for, we will check for the result.

e. Hover the cursor over the Start icon. Little shapes appear around the outside.

- Draw an arrow from the **Start** icon to the **Initialize** sequence.
- Draw another arrow from the **Initialize** sequence to the **Flow Decision**.



Step 7: True or False

The output of the Flow Decision is a True or False branch. You can change the labels of the a True or False branches; however, regardless of the labels, the condition is either met or not met. In our case, the condition is not met if the activity is not the Drop Activity or if the update did not close the status on that activity.

a. Drag a **LogLine** activity to the right and slightly below the Flow Decision.

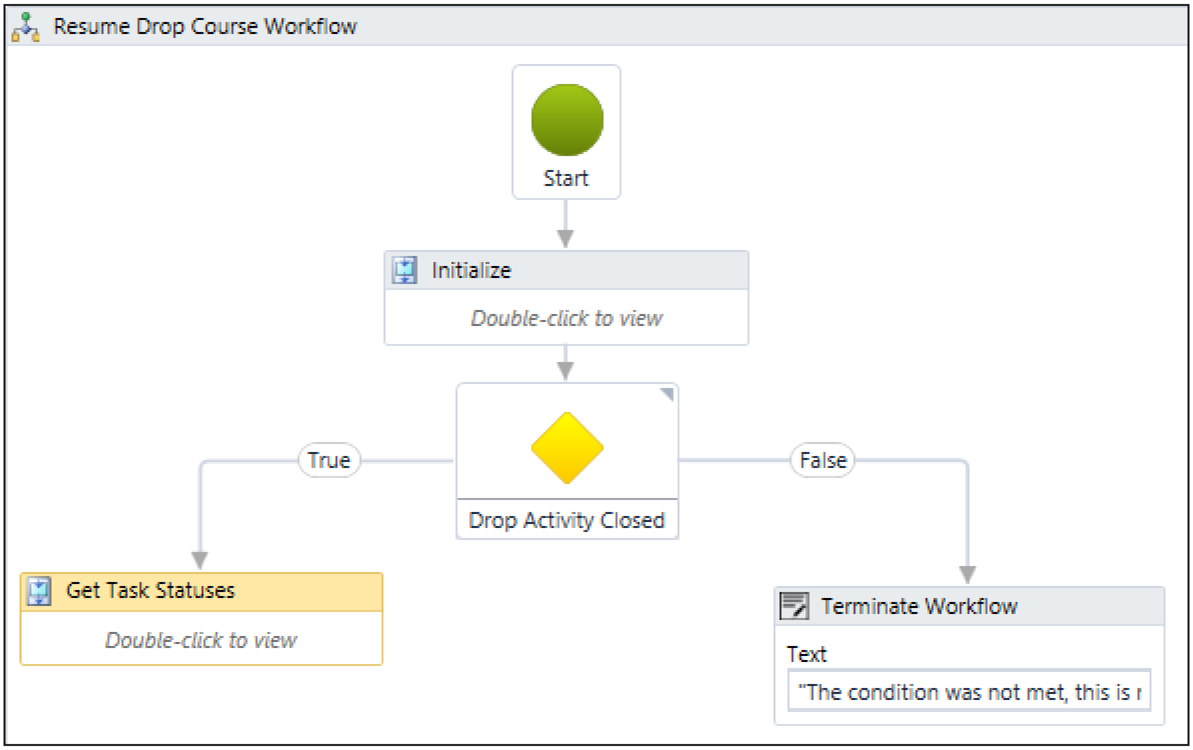
b. In the Properties pane, change the DisplayName to **Terminate Workflow**.

c. In the Text field, type: **"The condition was not met, this is not a Drop Activity"**.

d. Drag a **Sequence** to the left and slightly below the Flow Decision.

e. In the Properties pane, change the DisplayName to **Get Task Statuses**.

f. Connect the Flow Decision to each of the sequences.



Step 8: Get the Task Statuses

a. Double-click the **Get Task Statuses** sequence.

b. Drag two **LookupReferenceItem** activities and a **LogLine** activity into this sequence.

c. In the first LookupReferenceItem, specify the following:

DisplayName:	**Activity Result Approved**
Reference Item Type:	**Task Result**
Reference Item:	**ActivityResultApproved** (This is one of the variables created above.)

d. In the second LookupReferenceItem, specify the following:

DisplayName:	**Activity Result Denied**
Reference Item Type:	**Task Result**
Reference Item:	**ActivityResultDenied** (This is one of the variables created above.)

e. In the Text field of the LogLine activity, type:

"The ApprovalID is " + ActivityResultApproved.Id.ToString + " the DeniedId is " + ActivityResultDenied.Id.ToString

f. In the Properties pane of the LogLine activity, change the DisplayName to **Log the Result Ids**.

Step 9: More Flow Decisions

a. Drag **two more Flow Decisions** into the workflow.

b. Connect the **Get Task Statuses** sequence to the first Flow Decision.

c. In the Properties pane, change the DisplayName to **Approved**.

d. Connect the **False** line of the *Approved* decision to the top of the **Denied** Flow Decision.

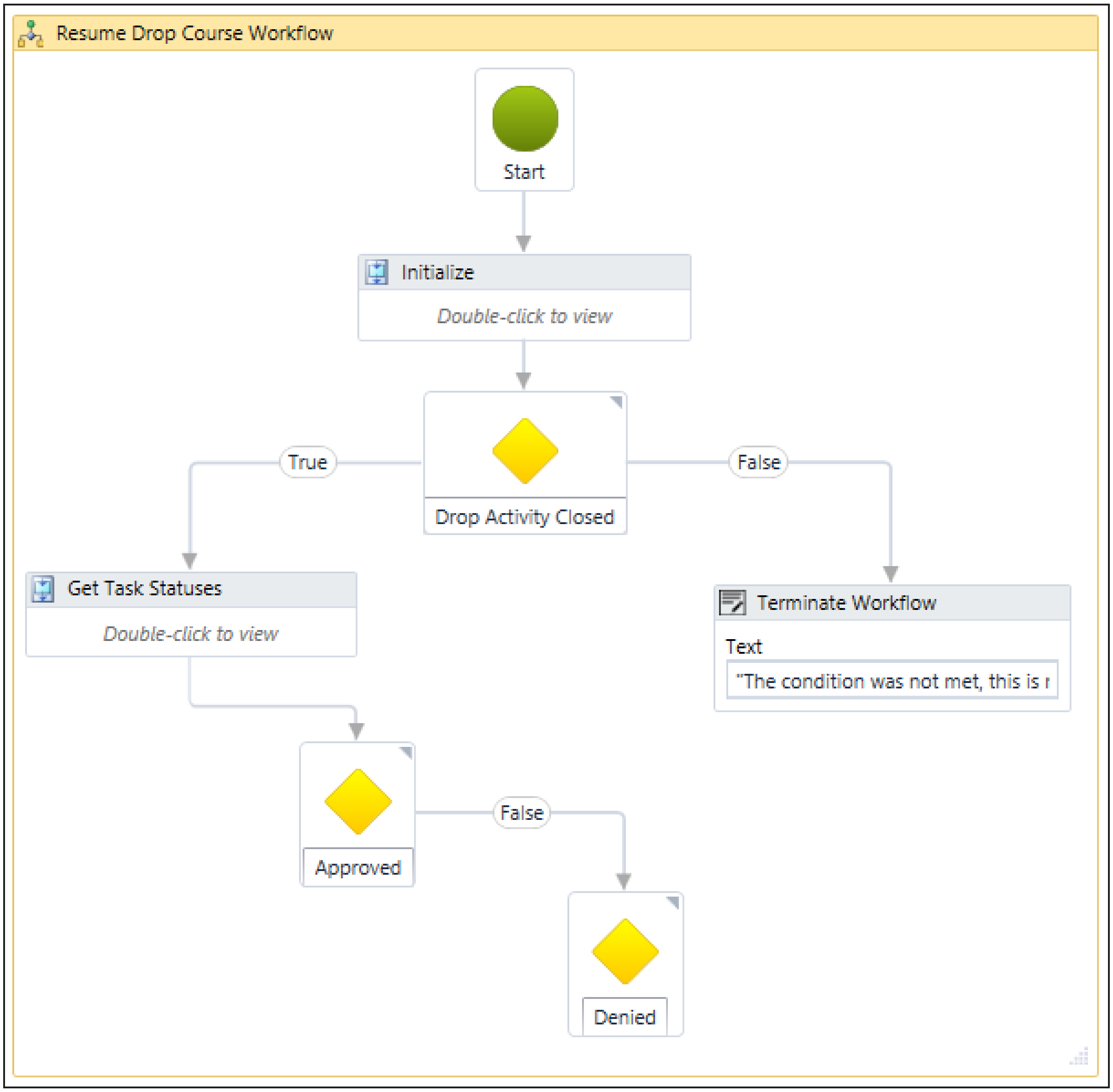
e. In the Properties pane, change the DisplayName to **Denied**.

f. Set the Condition field for the *Approved* decision to:

   entity.TaskResultId.Value = ActivityResultApproved.Id

g. Set the Condition field for the *Denied* decision to:

   entity.TaskResultId.Value = ActivityResultDenied.Id



Step 10: BookmarkName

a. Drag the **Assign** activity near and below the True side of the Approved decision.

   When the Approved decision goes down the True path, we are going to set the value of the variable BookmarkName to Approved. Remember, we set the default value to Denied (see step 3).

b. Drag a **LogLine** activity below the Assign activity.

c. In the Text field, type: **"The Bookmark name is " + BookmarkName**

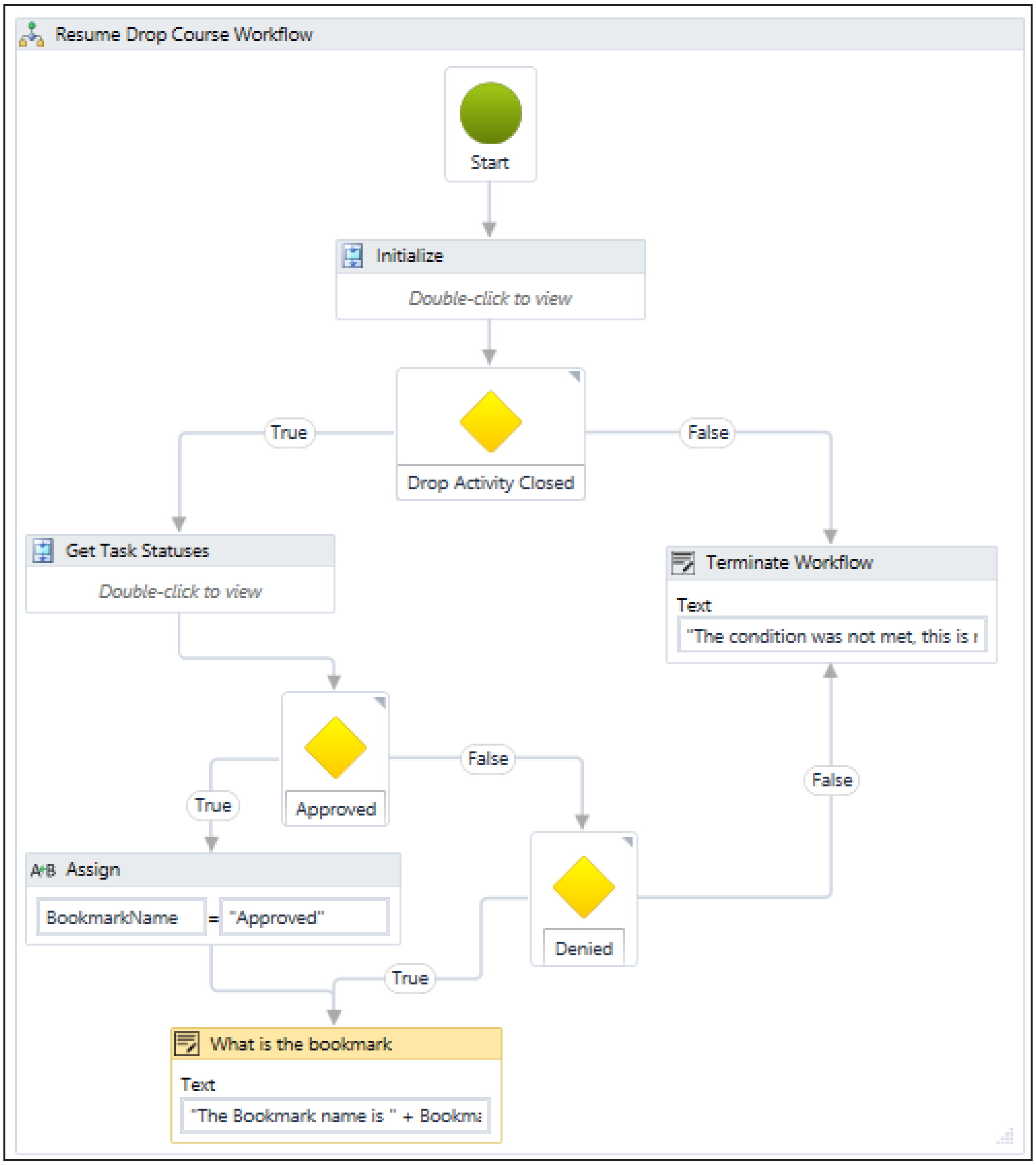
d. In the Properties pane of the LogLine activity, change the DisplayNameto **What is the bookmark**.

e. Connect the **Approved** True line to the **Assign** activity.

f. Connect the **Assign** activity to the **LogLine** activity.

g. Connect the **False** line of the Denied decision to the **Terminate** workflow activity.

h. Connect the **True** line to the **What is the bookmark** LogLine activity.



Step 11: Resume the sleeping workflow

a. Drag another **Sequence** into the workflow.

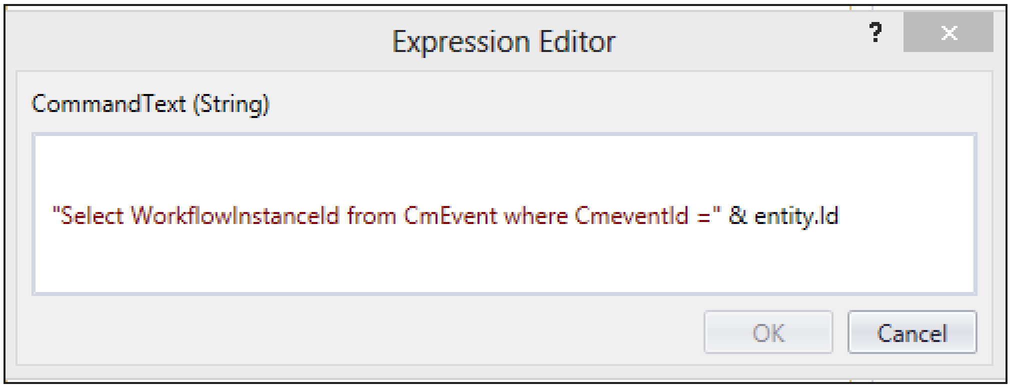
b. In the Properties pane, change the DisplayName to **Kick Off the Persisted Workflow**.

We need to get the WorkflowInstanceId and resume the bookmark that is waiting in our Pick Branch from our long running workflow.

c. Drag the **ExecuteDataReader** activity into the sequence.

d. In the Command field, type:

"Select WorkflowInstanceId from CmEvent where CmeventId =" & entity.Id



"Select WorkflowInstanceId from CmEvent where CmeventId =" & entity.Id

Step 12: Assign the WorkflowInstanceId to the GUID variable

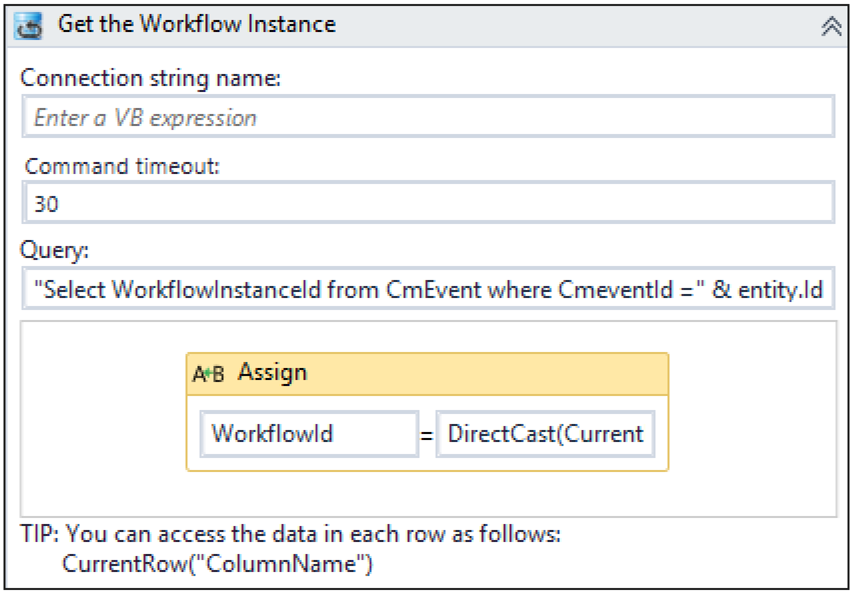
a. Drag the **Assign** activity into the **Get the Workflow Instance** activity.

b. In the *To* property, type **WorkflowId**. This is the GUID variable we created earlier.

c. In the *Value* property, type: **DirectCast(CurrentRow("WorkflowInstanceId"),GUID)**

Results that come from the ExecuteDataReader activity are always strings, but we need a GUID. Therefore, we are casting the result into the correct data type.

Important: It is critical that the spelling inside CurrentRow() is exactly as it is in our SELECT statement. Otherwise, the string to string comparison will fail.



Get the Workflow Instance

Connection string name: *Enter a VB expression*

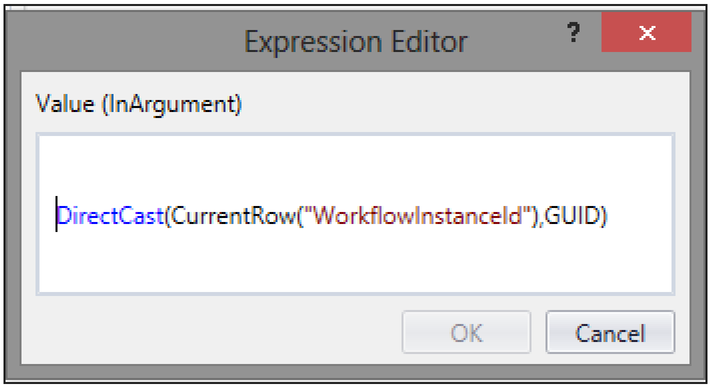
Command timeout: 30

Query: "Select WorkflowInstanceId from CmEvent where CmeventId =" & entity.Id

Assign

WorkflowId = DirectCast(Current

TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")



Expression Editor

Value (InArgument)

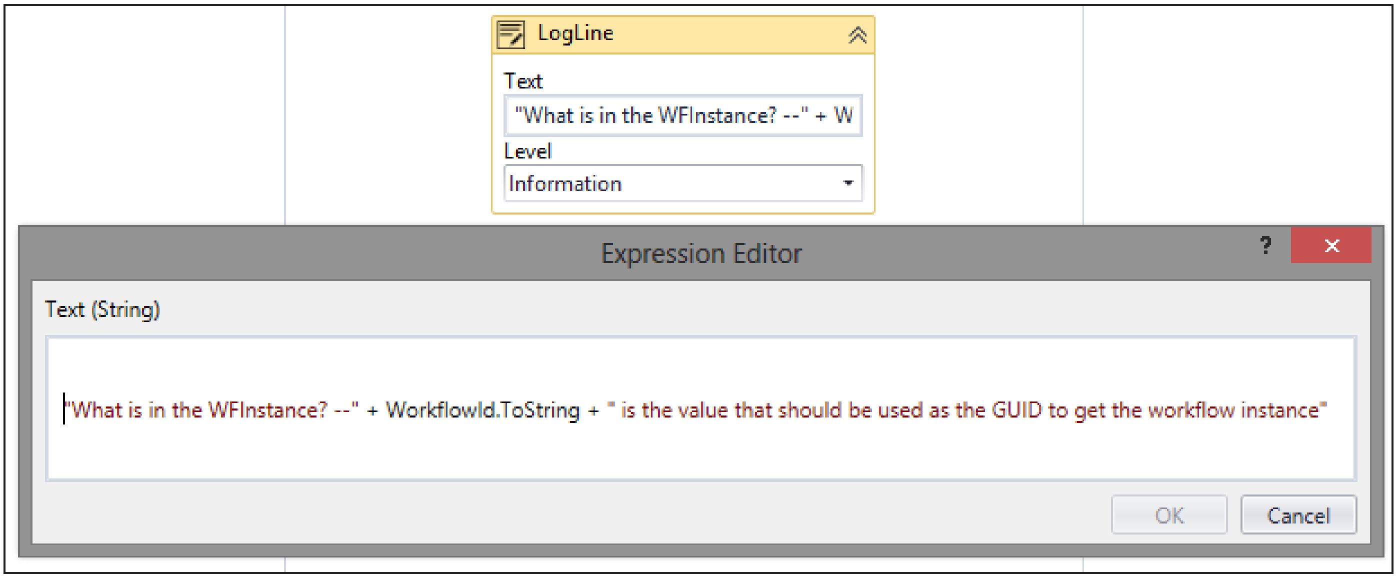
```
DirectCast(CurrentRow("WorkflowInstanceId"),GUID)
```

Step 13: Log the output of the logic before resuming the workflow

a. Drag a **LogLine** activity under the Get the Workflow Instance activity.

b. In the Text field, type:

"**What is in the WFInstance? --" + WorkflowId.ToString + " is the value that should be used as the GUID to get the workflow instance"**



LogLine

Text

"What is in the WFInstance? --" + W

Level

Information

Text (String)

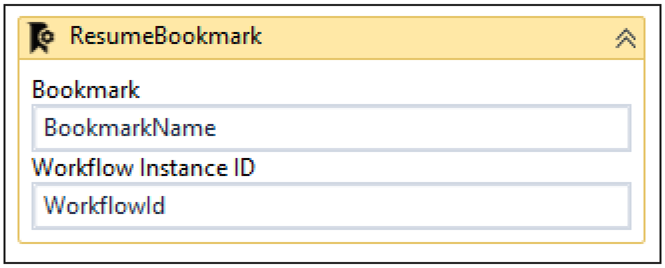
```
"What is in the WFInstance? --" + WorkflowId.ToString + " is the value that should be used as the GUID to get the workflow instance"
```

Step 14: Resume the workflow

a. In the Toolbox under Cmc.Core.Workflow.Activities, find the **ResumeBookmark** activity and drag it under the LogLine activity.

b. We have the BookmarkName set by our decision logic and we now have our WorkflowId assigned to the instance we are looking for.

- In the Bookmark field, add the **BookmarkName** variable.
- In the Workflow Instance Id field, add the **WorkflowId** variable.



ResumeBookmark

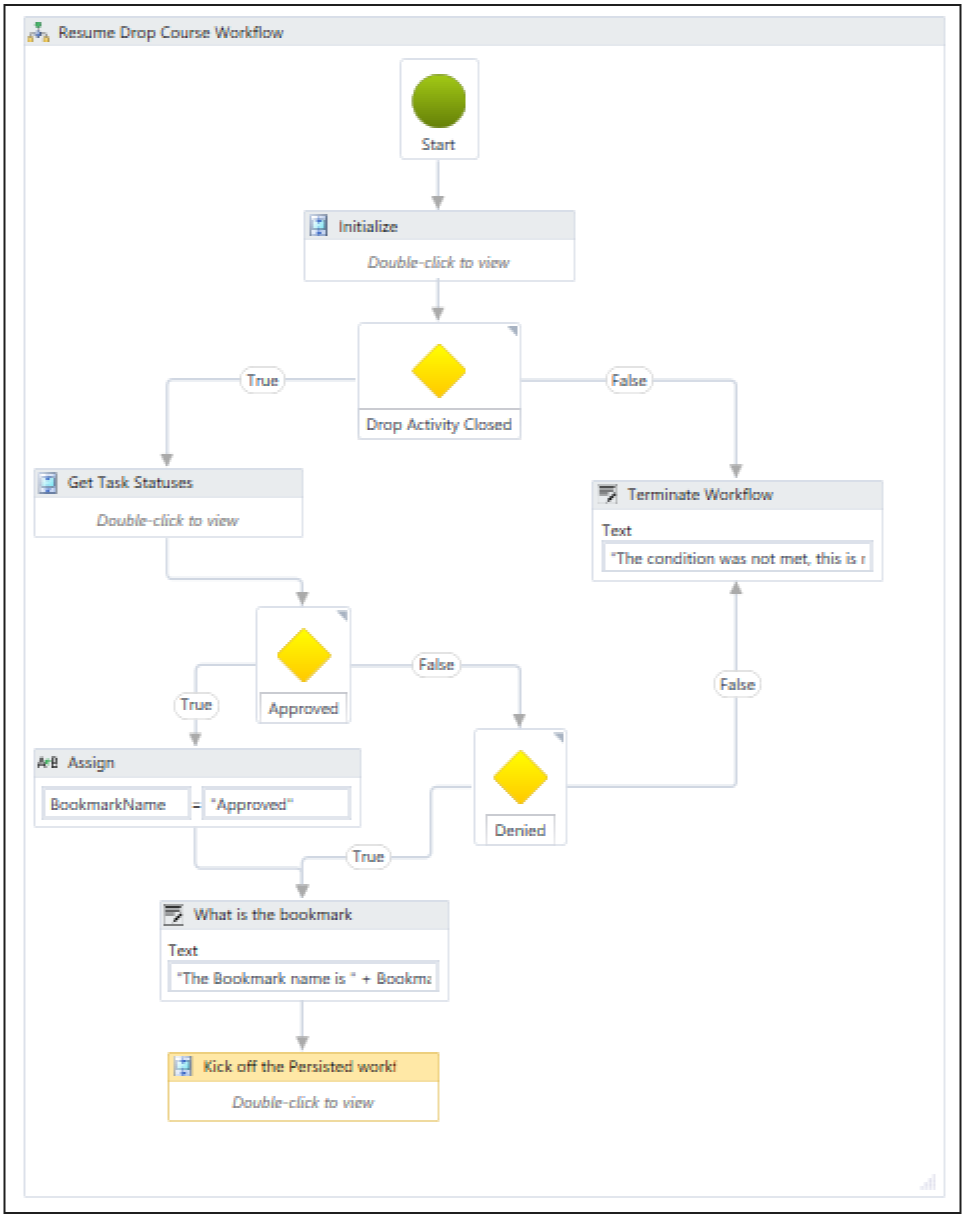
Bookmark

BookmarkName

Workflow Instance ID

WorkflowId

c. Connect the **What is the bookmark** LogLine activity to the **Kick off the Persisted Workflow**.

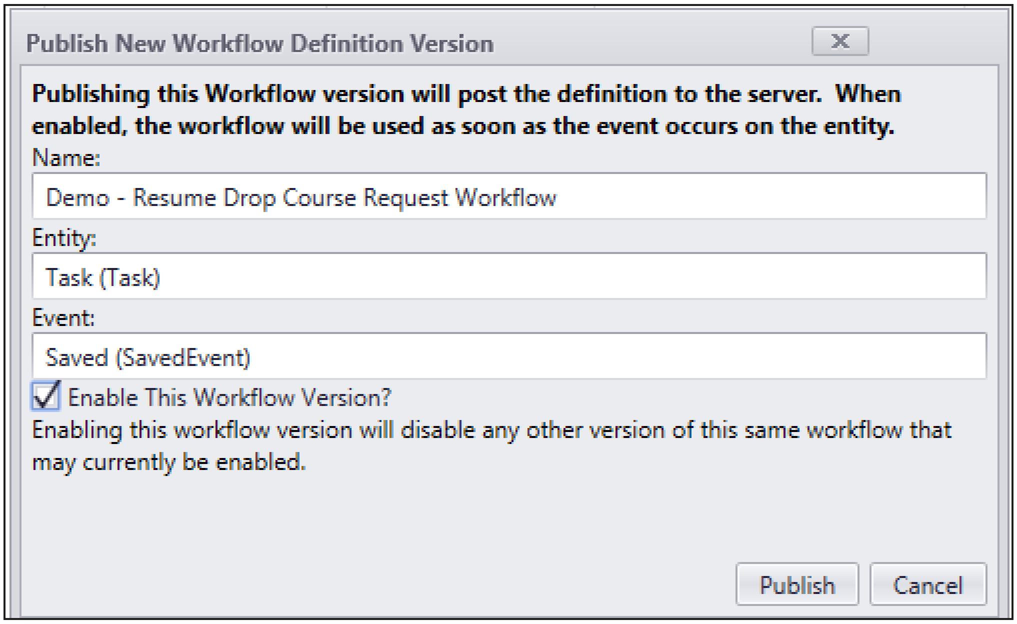


Step 15: Save and publish the workflow

a. Check your workflow. Scroll through the workflow or use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.

b. Click **Publish**. The Publish Workflow Definition Version window is displayed.

c. Select **Enable This Workflow Version**.



Publish New Workflow Definition Version

Publishing this Workflow version will post the definition to the server. When enabled, the workflow will be used as soon as the event occurs on the entity.

Name:

Demo - Resume Drop Course Request Workflow

Entity:

Task (Task)

Event:

Saved (SavedEvent)

☑ Enable This Workflow Version?

Enabling this workflow version will disable any other version of this same workflow that may currently be enabled.

d. Click **Publish**, then **Cancel** to close the publisher window.

Continue with Test the Workflow Sequence.

Test the Workflow Sequence

Before you test the long running workflow sequence:

- Make sure the Service Module Host service is running.
- Navigate to the location of your log files for the Service Module Host. There should be one file with today's date.

Step 1: Add Contact Manager Activity in Anthology Student

a. Launch Anthology Student.

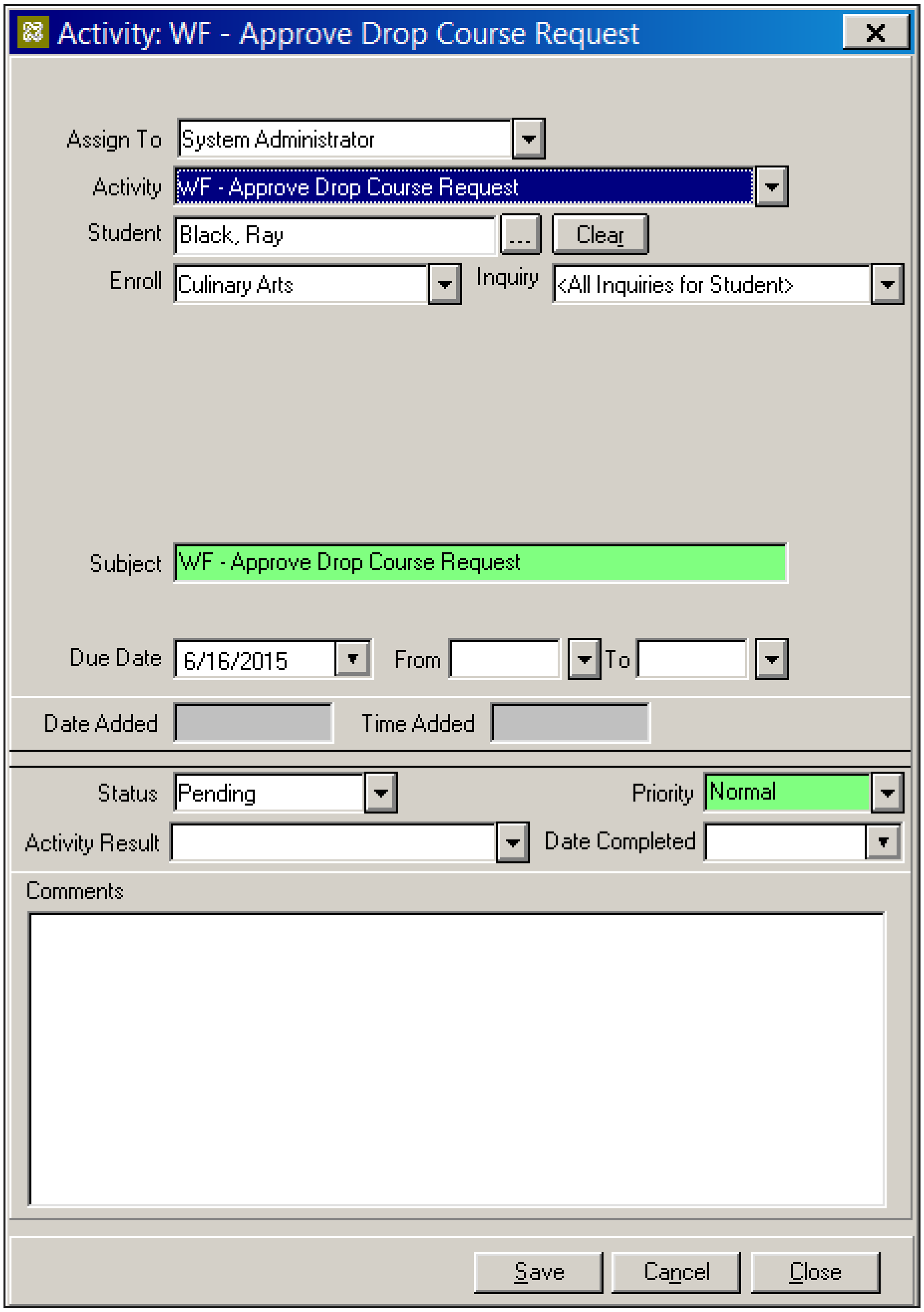
b. Find a student.

c. Open the student's **Activities** folder.

d. Navigate to **View > Contact Manager > Activities**.

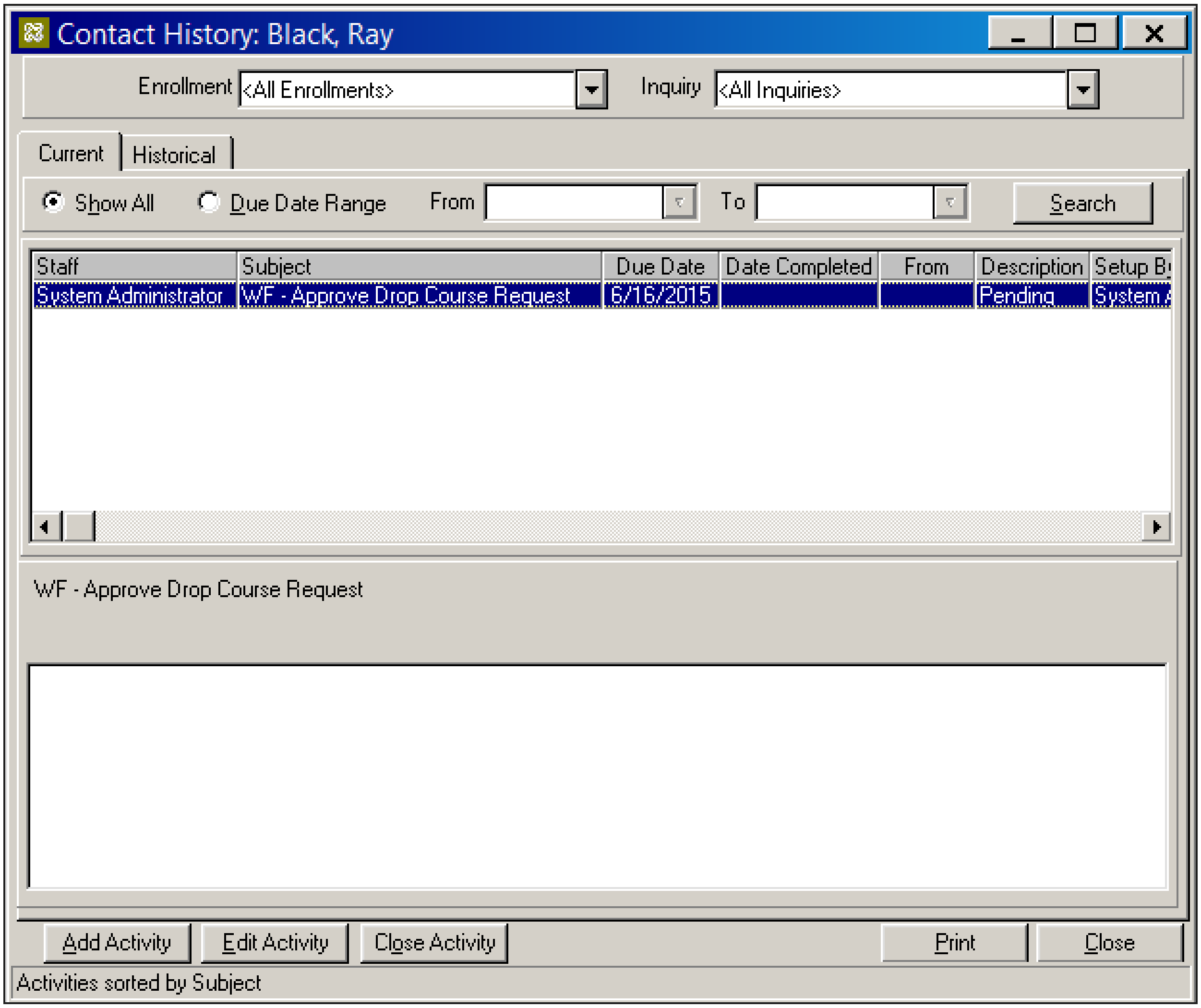
e. Add the activity **WF - Approve Drop Course Request**.

f. Click **Save**.



Activity: WF - Approve Drop Course Request

Assign To	System Administrator
Activity	WF - Approve Drop Course Request
Student	Black, Ray
Enroll	Culinary Arts
Inquiry	<All Inquiries for Student>
Subject	WF - Approve Drop Course Request
Due Date	6/16/2015
From	
To	
Date Added	
Time Added	
Status	Pending
Priority	Normal
Activity Result	
Date Completed	
Comments	



Enrollment <All Enrollments>

Inquiry <All Inquiries>

● Show All ○ Due Date Range

Staff	Subject	Due Date	Date Completed	From	Description	Setup B
System Administrator	WF - Approve Drop Course Request	6/16/2015			Pending	System

WF - Approve Drop Course Request

Activities sorted by Subject

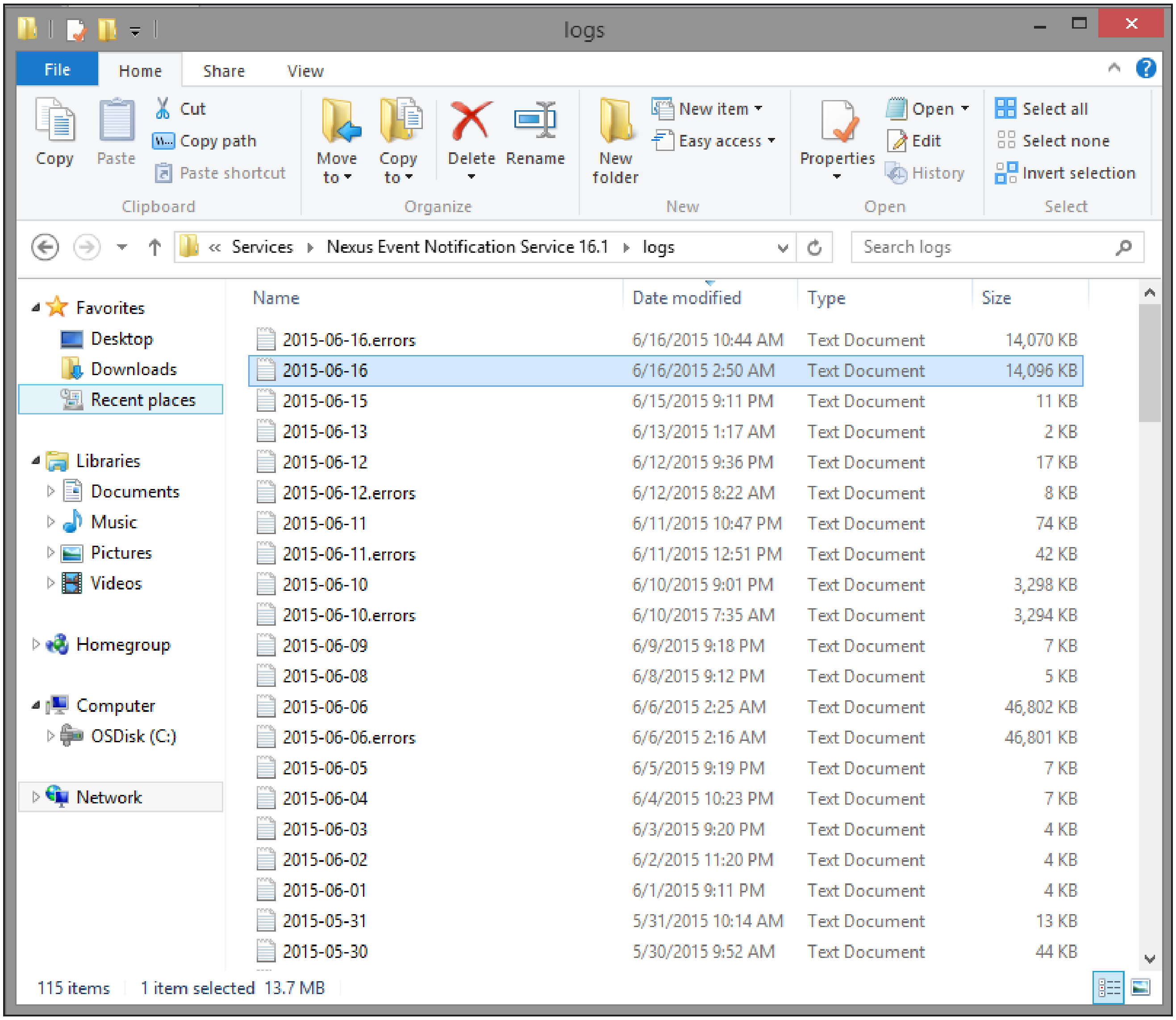
This will raise the Task Saved event and should kick off the long running workflow named *Demo – How to use a long running workflow*.

After you click Save, you can open your log file. It should have logged our all LogLine activities contained ion the workflow sequence.

Step 2: Check the log and verify the update

a. On the server running the workflow, navigate to `Services\Nexus Event Notification Service 16.1\logs`.

b. Double-click the most recent log file.



« Services ▸ Nexus Event Notification Service 16.1 ▸ logs

Name	Date modified	Type	Size
2015-06-16.errors	6/16/2015 10:44 AM	Text Document	14,070 KB
2015-06-16	6/16/2015 2:50 AM	Text Document	14,096 KB
2015-06-15	6/15/2015 9:11 PM	Text Document	11 KB
2015-06-13	6/13/2015 1:17 AM	Text Document	2 KB
2015-06-12	6/12/2015 9:36 PM	Text Document	17 KB
2015-06-12.errors	6/12/2015 8:22 AM	Text Document	8 KB
2015-06-11	6/11/2015 10:47 PM	Text Document	74 KB
2015-06-11.errors	6/11/2015 12:51 PM	Text Document	42 KB
2015-06-10	6/10/2015 9:01 PM	Text Document	3,298 KB
2015-06-10.errors	6/10/2015 7:35 AM	Text Document	3,294 KB
2015-06-09	6/9/2015 9:18 PM	Text Document	7 KB
2015-06-08	6/8/2015 9:12 PM	Text Document	5 KB
2015-06-06	6/6/2015 2:25 AM	Text Document	46,802 KB
2015-06-06.errors	6/6/2015 2:16 AM	Text Document	46,801 KB
2015-06-05	6/5/2015 9:19 PM	Text Document	7 KB
2015-06-04	6/4/2015 10:23 PM	Text Document	7 KB
2015-06-03	6/3/2015 9:20 PM	Text Document	4 KB
2015-06-02	6/2/2015 11:20 PM	Text Document	4 KB
2015-06-01	6/1/2015 9:11 PM	Text Document	4 KB
2015-05-31	5/31/2015 10:14 AM	Text Document	13 KB
2015-05-30	5/30/2015 9:52 AM	Text Document	44 KB

115 items 1 item selected 13.7 MB

c. The log shows that two workflows were triggered by a Task Saved event:

- The first workflow checks if it the Drop Course activity is added, saves the WorkflowInstanceId, and then goes to sleep.
- The second workflow waits for the activity to close with a result.

Because we added the activity and did not close it, the second workflow indicates that the condition is not met and stops.

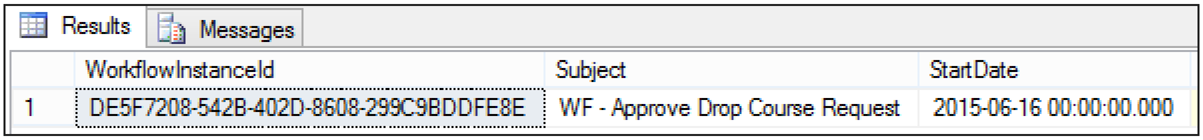
```
Queue, Type: //Cmc/SSBMessage_CmEvent_Saved_Notification

2015-06-16 14:26:04.5788  14 Trace                         Cmc.Core.Eventing.SavedEvent

Raising event 'Saved' on type 'Task' - Entering
```

```
2015-06-16 14:26:04.5788  14 Trace                          Cmc.Core.Eventing.SavedEvent

Executing handler 'Cmc.Core.Workflow.WorkflowEventHandler`2[Cmc.Core.Eventing.SavedEvent,Cmc.Nexus.Crm.Task]' - Entering

2015-06-16 14:26:04.5944  14 Debug                          Cmc.Core.Workflow.WorkflowEngine

Running workflow de5f7208-542b-402d-8608-299c9bddfe8e

2015-06-16 14:26:04.6256  86 Info                           Cmc.Core.Workflow.Activities.LogLine

Starting Long Running Workflow Example - The Activity we are looking for is WF - Approve Drop Course Request with the TaskTemplateId = 272

2015-06-16 14:26:04.6412  19 Info                           Cmc.Core.Workflow.Activities.LogLine

The workflow instance - de5f7208-542b-402d-8608-299c9bddfe8e was added to the CmEventID-16356

2015-06-16 14:26:04.6568  34 Info                           Cmc.Core.Workflow.Activities.LogLine

Pausing the workflow - awaiting approver result

2015-06-16 14:26:04.6568  15 Trace                          Cmc.Nexus.Utility.ServiceBroker.ServiceModule.ServiceBrokerServiceModule 15: New Message From Queue, Type: //Cmc/SSBMessage_CmEvent_Saved_Notification

2015-06-16 14:26:04.6568  14 Debug                          Cmc.Core.Workflow.WorkflowEngine

Done running workflow de5f7208-542b-402d-8608-299c9bddfe8e

2015-06-16 14:26:04.6724  14 Debug                          Cmc.Core.Workflow.WorkflowEngine

Running workflow cc8dced3-76ff-4906-85fc-46d3db755789

2015-06-16 14:26:04.6880  25 Info                           Cmc.Core.Workflow.Activities.LogLine

Check event for conditions - the CmEventId is -16356 the StatusId is - 1 the ResultId is - 0

2015-06-16 14:26:04.7036  77 Info                           Cmc.Core.Workflow.Activities.LogLine

The condition was not met, this is not a Drop Activity

2015-06-16 14:26:04.7036  14 Debug                          Cmc.Core.Workflow.WorkflowEngine

Done running workflow cc8dced3-76ff-4906-85fc-46d3db755789
```

Step 3: Check the database and verify the update

Use the following SQL statement to verify that the GUID was saved to the CmEvent table:

Select Top 1 WorkflowInstanceId, [Subject], * from CmEvent order by DateAdded Desc

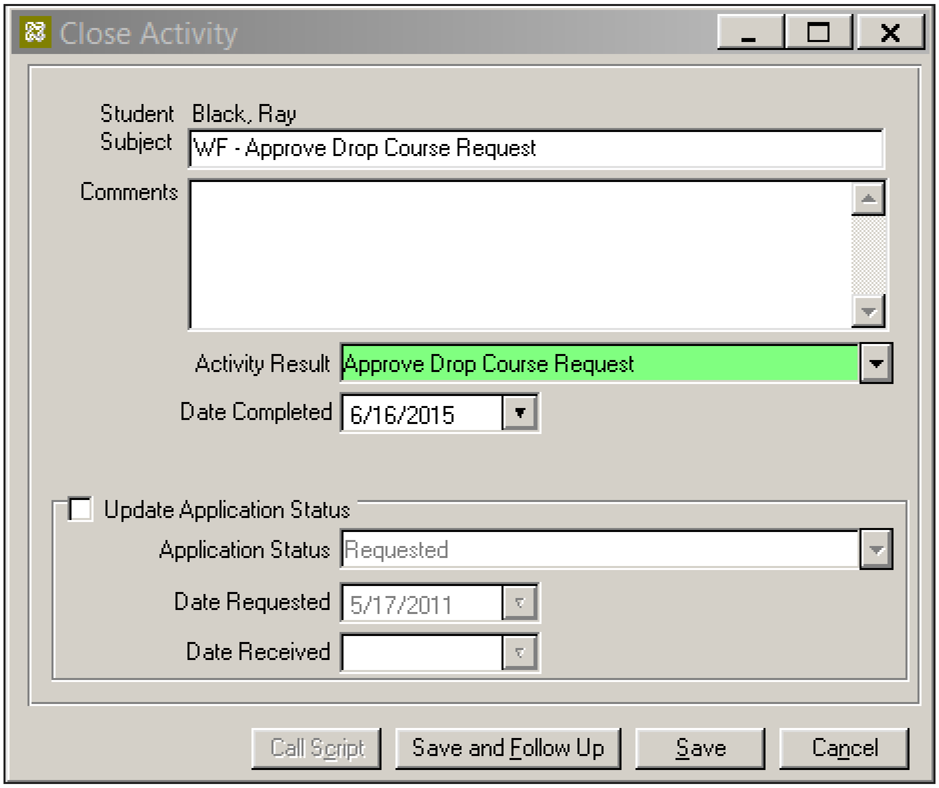


	WorkflowInstanceId	Subject	StartDate
1	DE5F7208-542B-402D-8608-299C9BDDFE8E	WF - Approve Drop Course Request	2015-06-16 00:00:00.000

Step 4: Resume the workflow

a. From the student's **Activities** folder, find the activity we just added and click **Close Activity**.

b. Select **Approve Drop Course Request** as the result.



Student Black, Ray

Subject WF - Approve Drop Course Request

Comments

Activity Result Approve Drop Course Request

Date Completed 6/16/2015

□ Update Application Status

Application Status Requested

Date Requested 5/17/2011

Date Received

This will resume the Approved Process Pick Branch.

c. Check the log file again.

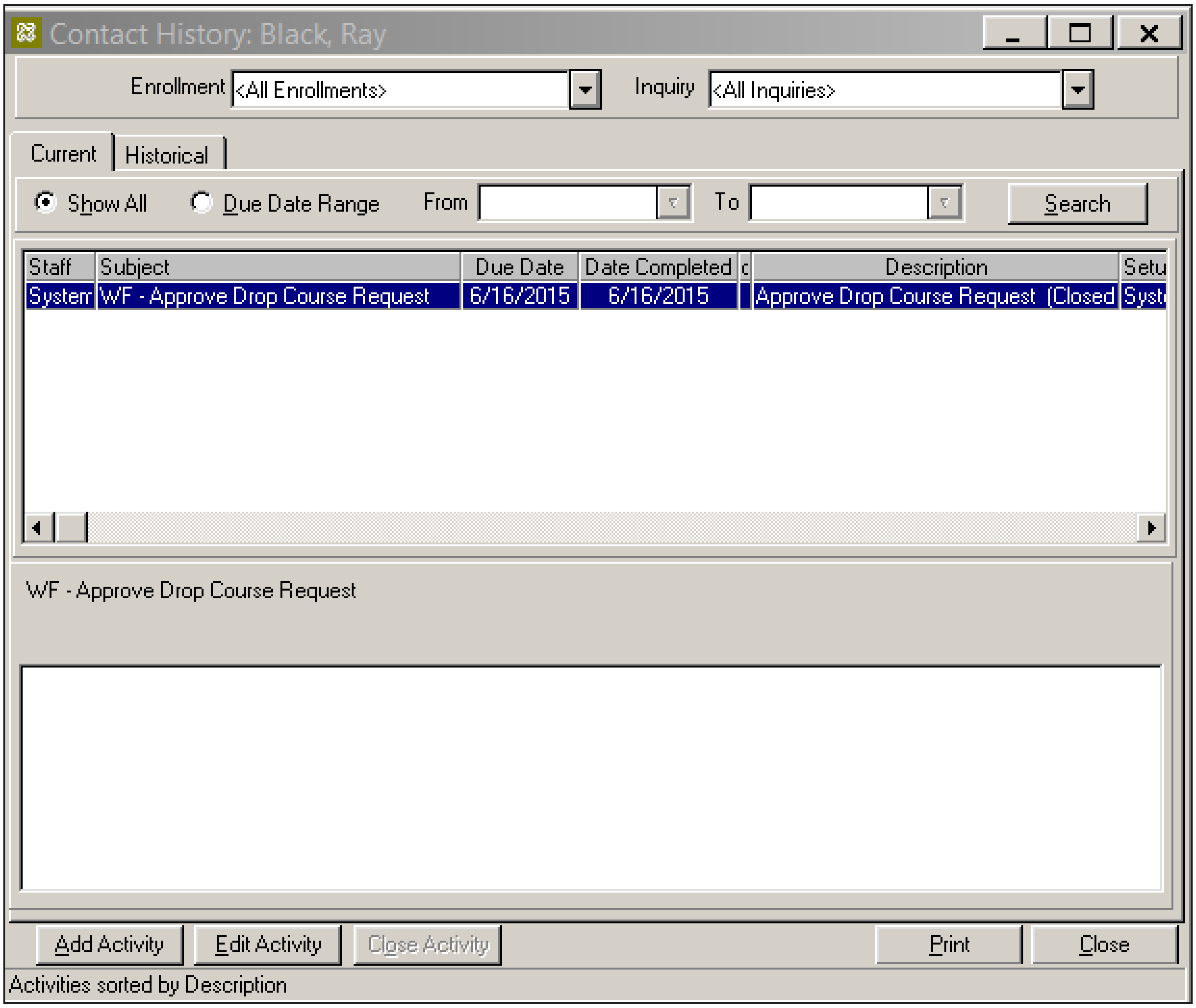
```
[Cmc.Core.Eventing.SavedEvent,Cmc.Nexus.Person]' - Entering

2015-06-16 14:38:38.5729  15 Debug                                  Cmc.Core.Workflow.WorkflowEngine
```

```
Running workflow f0a879e3-2d61-48ba-bd60-68a2701eff2a

2015-06-16 14:38:38.5729  13 Debug                          Cmc.Core.Work-
flow.WorkflowEngine

Running workflow 7aad21e1-d6e2-4307-9916-0636e449977a

2015-06-16 14:38:38.6197  46 Info                           Cmc.Core.Work-
flow.Activities.LogLine

Starting Long Running Workflow Example - The Activity we are looking for is
WF - Approve Drop Course Request with the TaskTemplateId = 272

2015-06-16 14:38:38.6353  25 Info                           Cmc.Core.Work-
flow.Activities.LogLine

Condition not met, terminating workflow

2015-06-16 14:38:38.6353  74 Info                           Cmc.Core.Work-
flow.Activities.LogLine

Pausing the workflow - awaiting approver result

2015-06-16 14:38:38.6353  15 Debug                          Cmc.Core.Work-
flow.WorkflowEngine

Done running workflow f0a879e3-2d61-48ba-bd60-68a2701eff2a

2015-06-16 14:38:38.6509  15 Debug                          Cmc.Core.Work-
flow.WorkflowEngine

Running workflow d29cb1c9-337f-4a8c-8b20-2bed86eaf9a9

2015-06-16 14:38:38.6665  13 Debug                          Cmc.Core.Work-
flow.WorkflowEngine

Done running workflow 7aad21e1-d6e2-4307-9916-0636e449977a

2015-06-16 14:38:38.6665  13 Trace                          Cmc.Core.Event-
ing.SavedEvent

Executing handler 'Cmc.Core.Workflow.WorkflowEventHandler`2[Cmc.Core.Event-
ing.SavedEvent,Cmc.Nexus.Person]' - Exiting

2015-06-16 14:38:38.6665  13 Trace                          Cmc.Core.Event-
ing.SavedEvent

Raising event 'Saved' on type 'Person' - Exiting

2015-06-16 14:38:38.6821  25 Info                           Cmc.Core.Work-
flow.Activities.LogLine

Check event for conditions - the CmEventId is -16356 the StatusId is - 2 the Res-
ultId is - 21

2015-06-16 14:38:38.6977  95 Info                           Cmc.Core.Work-
flow.Activities.LogLine

The ApprovalID is 21 the DeniedId is 22
```

```
2015-06-16 14:38:38.6977  95 Info                         Cmc.Core.Workflow.Activities.LogLine

The Bookmark name is Approved

2015-06-16 14:38:38.7133  46 Info                         Cmc.Core.Workflow.Activities.LogLine

What is in the WFInstance? --de5f7208-542b-402d-8608-299c9bddfe8e is the value
that should be used as the GUID to get the workflow instance

2015-06-16 14:38:38.7445  44 Debug                        Cmc.Core.Workflow.WorkflowEngine

Running workflow de5f7208-542b-402d-8608-299c9bddfe8e

2015-06-16 14:38:38.7445  44 Debug                        Cmc.Core.Workflow.WorkflowEngine

Done running workflow de5f7208-542b-402d-8608-299c9bddfe8e

2015-06-16 14:38:38.7445  95 Info                         Cmc.Core.Workflow.Activities.LogLine

The Request was Approved

2015-06-16 14:38:38.7445  15 Debug                        Cmc.Core.Workflow.WorkflowEngine

Done running workflow d29cb1c9-337f-4a8c-8b20-2bed86eaf9a9

2015-06-16 14:38:38.7445  15 Trace                        Cmc.Core.Eventing.SavedEvent

Executing handler 'Cmc.Core.Workflow.WorkflowEventHandler`2[Cmc.Core.Eventing.SavedEvent,Cmc.Nexus.Crm.Task]' - Exiting
```

Test Successful!

Check the Contact History In Anthology Student and verify that the Approve Drop Course Request activity is closed.



Contact History: Black, Ray

Enrollment <All Enrollments>

Inquiry <All Inquiries>

● Show All ○ Due Date Range From To

Staff	Subject	Due Date	Date Completed	c	Description	Setu
Systerr	WF - Approve Drop Course Request	6/16/2015	6/16/2015		Approve Drop Course Request (Closed	Syst

WF - Approve Drop Course Request

Activities sorted by Description

Populate Fields in a Forms Builder Form

When web forms are built with Forms Builder 1.x or 2.x, eventing and workflows can be used to gather data and push the data into a multi-step form as it transitions from one step to another. Eventing and workflow make it possible to return information to a user on a Forms Builder web form that is not part of the existing adapter. In this scenario, we will return all of the courses a student is currently registered in.

Scenario

We built a Forms Builder form that allows a student to drop a course. For the first page of the form we wanted to make sure we had correct contact information for the student as dropping a course is a retention red-flag. Once the student verified his or her information, we used a workflow with AddToDictionary<> activity to get the current list of courses that the student was registered in.

Prerequisites

- A Forms Builder form to request admission was created.
- A student has registered into current courses with a Portal account.

Procedure

Step 1: Create the Forms Builder fields

a. Launch Forms Builder Designer.

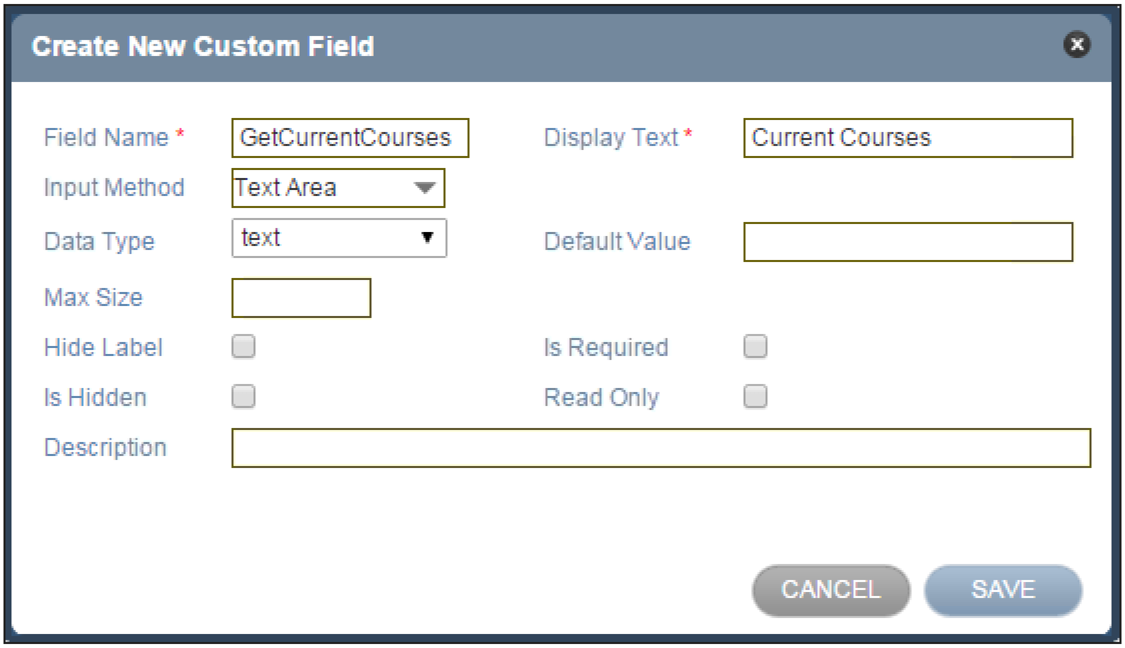
b. In the Fields tab, click **Add New Field**. The Create New Custom Field form is displayed.

c. In the Field Name field, type **GetCurrentCourses**.

d. In the Display Text field, type **Current Courses**.

e. In the Input Method field, select **Text Area**.

f. In the Data Type field, select **text**.



Create New Custom Field

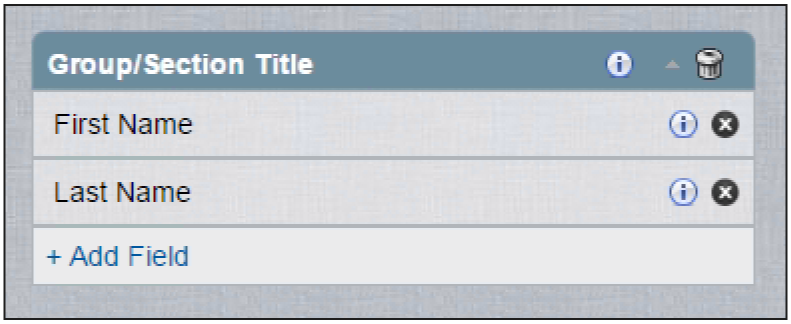
Field Name *	GetCurrentCourses	Display Text *	Current Courses
Input Method	Text Area		
Data Type	text	Default Value	
Max Size			
Hide Label	□	Is Required	□
Is Hidden	□	Read Only	□
Description			

g. Click **Save**.

Step 2: Create a simple form

a. From the Fields tab, search for **Name**.

b. Drag the **First Name** and **Last Name** fields onto the canvas.



Group/Section Title

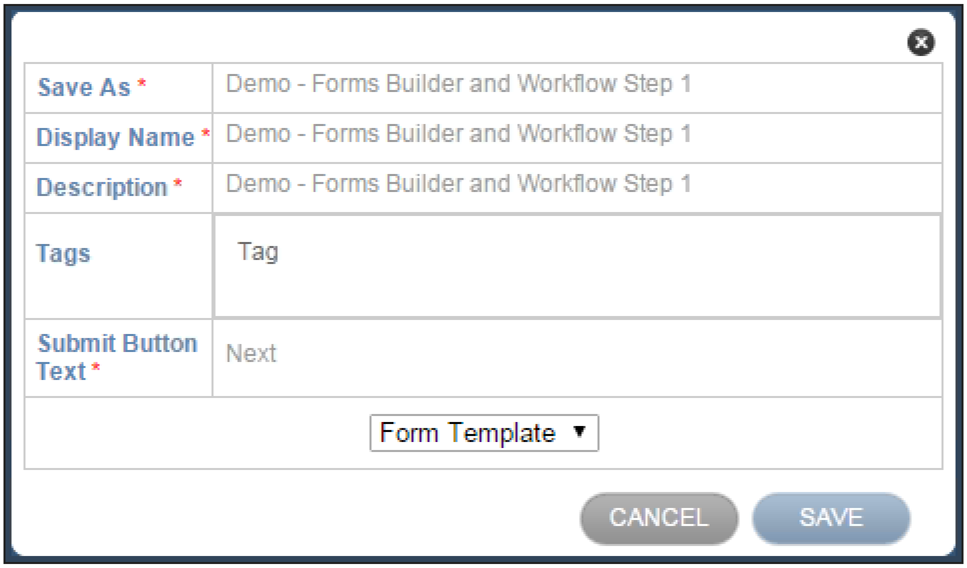
First Name

Last Name

Step 3: Save the Form Template

a. Click the **Save** button in the lower left corner.

b. Save as a **Form Template** named **Demo - Forms Builder and Workflow Step 1**.



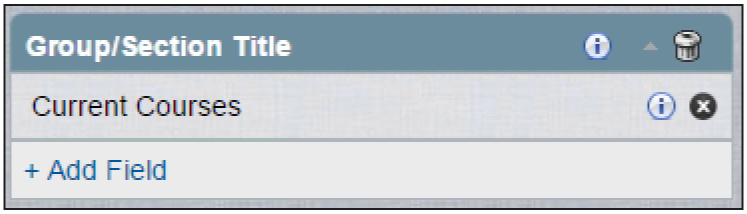
Save As *	Demo - Forms Builder and Workflow Step 1
Display Name *	Demo - Forms Builder and Workflow Step 1
Description *	Demo - Forms Builder and Workflow Step 1
Tags	Tag
Submit Button Text *	Next
	Form Template ▼

c. **Clear** the canvas.

Step 4: Drag the custom field on to the canvas.

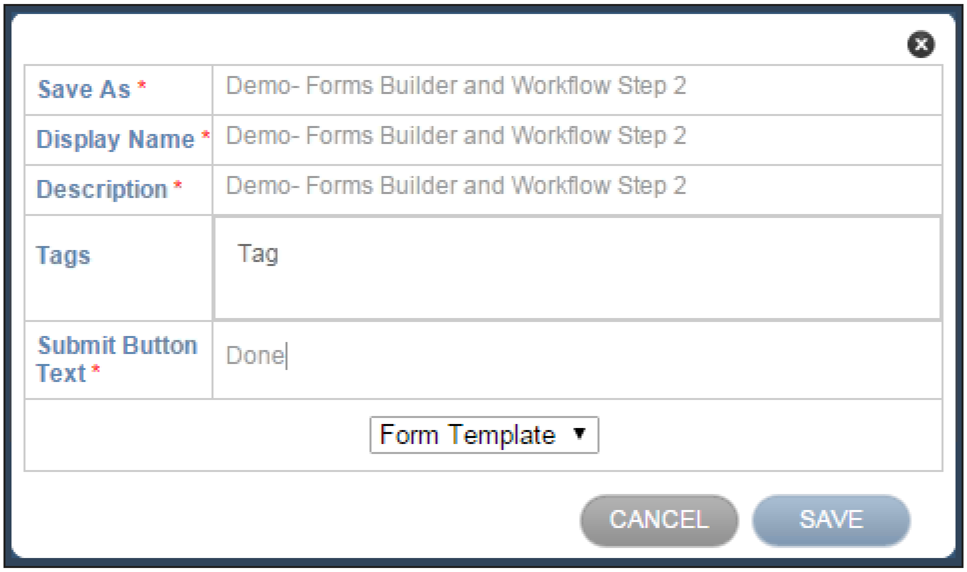
a. From the Fields tab, search for **courses**. The list of fields is filtered showing the custom field you created earlier.

b. Drag the **Current Courses** field onto the canvas.



c. Click **Save**.

d. Save as a **Form Template**.



Save As *	Demo- Forms Builder and Workflow Step 2
Display Name *	Demo- Forms Builder and Workflow Step 2
Description *	Demo- Forms Builder and Workflow Step 2
Tags	Tag
Submit Button Text *	Done

Form Template ▼

e. **Clear** the canvas.

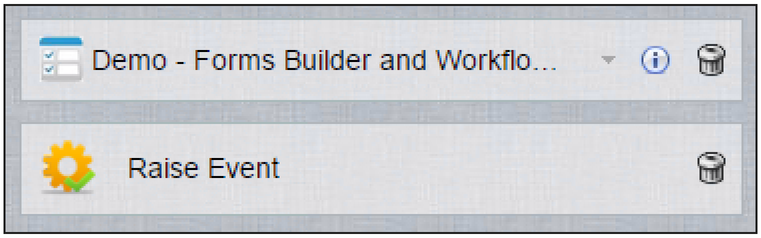
Step 5: Bring it together - Forms & Rules

a. In the Forms and Rules tab, search for **demo**.

b. Drag the form template named **Demo - Forms Builder and Workflow Step 1** on to the canvas.

c. Click the **Rules** tab

d. Drag the **Raise Event** rule under the Demo – Forms Builder and Workflow Step 1 form template.

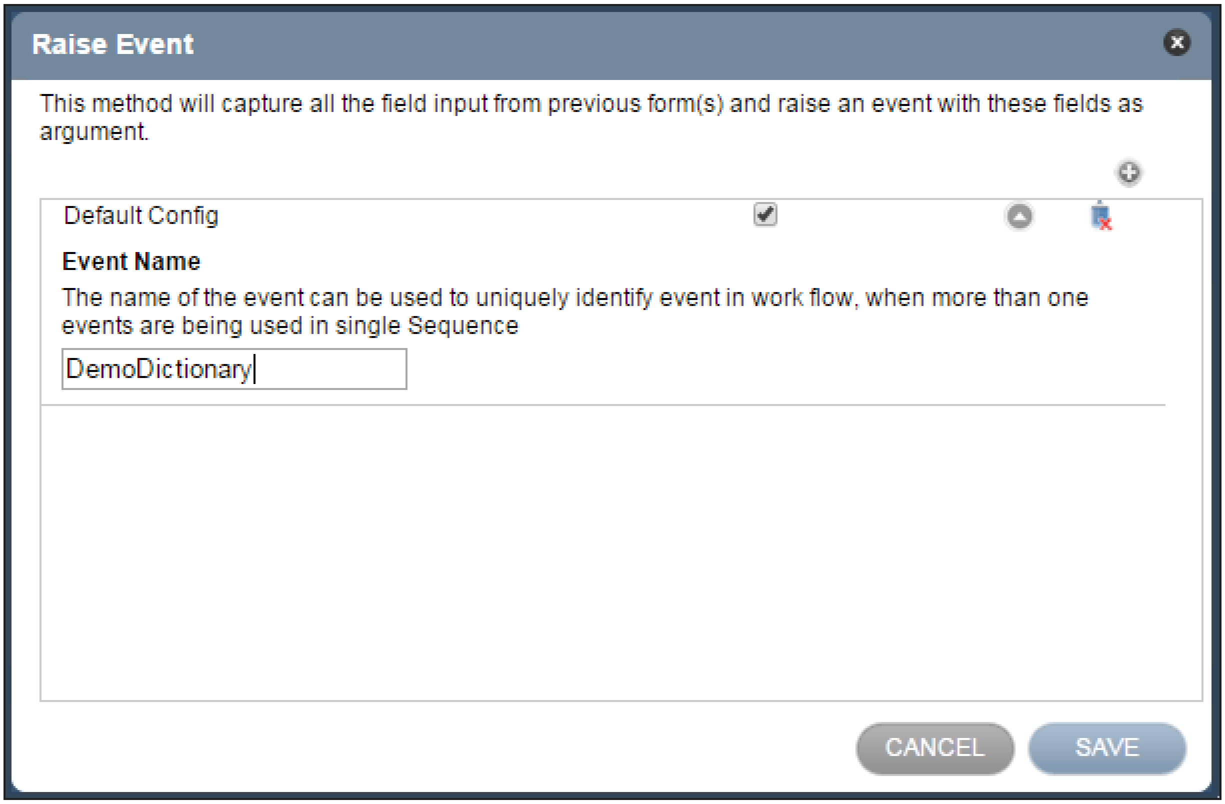


Demo - Forms Builder and Workflo...

Raise Event

e. When you drag the Raise Event rule onto the canvas, the default configuration requires that you name the event. This name will initiate the workflow.

Type **DemoDictionary** in the Event Name field, and click **Save**.



Raise Event

This method will capture all the field input from previous form(s) and raise an event with these fields as argument.

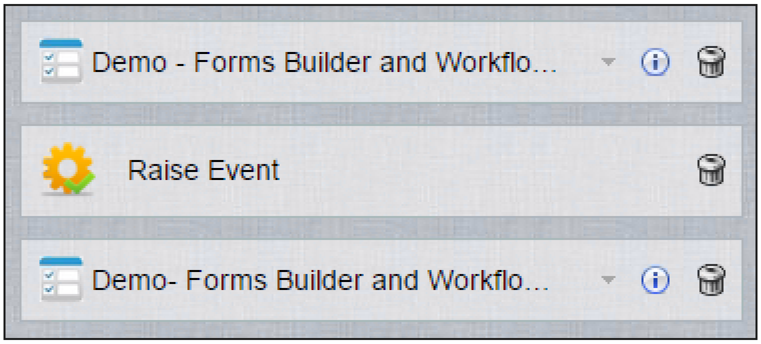
Default Config ☑

Event Name

The name of the event can be used to uniquely identify event in work flow, when more than one events are being used in single Sequence

DemoDictionary

f. Drag **Demo – Forms Builder and Workflow Step 2** on to the canvas under the rule.



Demo - Forms Builder and Workflo...

Raise Event

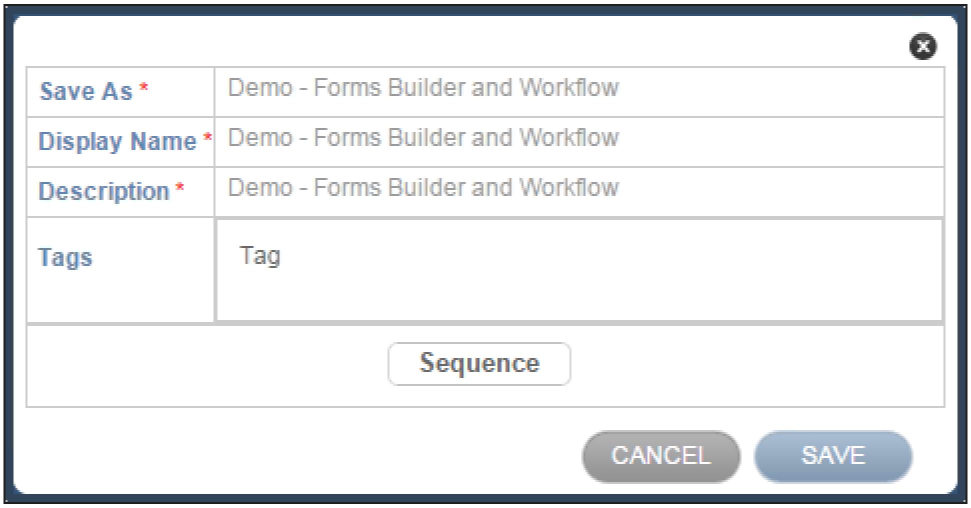
Demo- Forms Builder and Workflo...

g. Click **Save**.

Step 6: Save the Sequence

a. Fill out the **Save Sequence** form.

b. Click **Save**.



Save As * Demo - Forms Builder and Workflow

Display Name * Demo - Forms Builder and Workflow

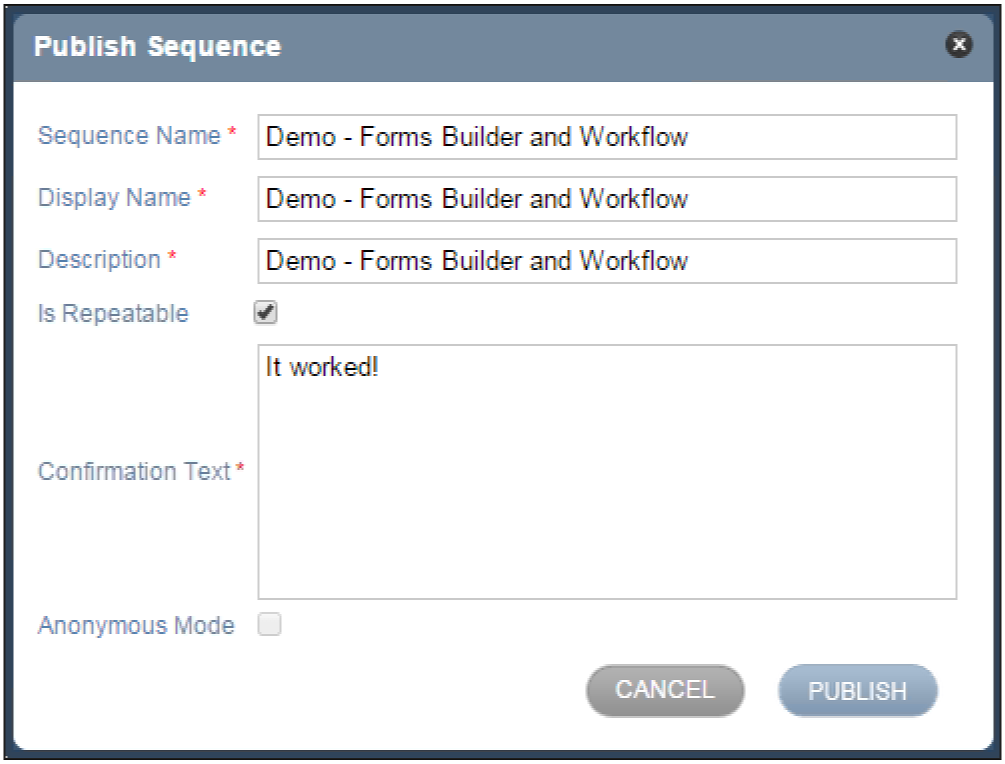
Description * Demo - Forms Builder and Workflow

Tags Tag

c. In the upper right hand side, click **Publish**.

d. Select the **Is Repeatable** check box and type **It Worked!** in the Confirmation Text field.

e. Click **Publish**.



Publish Sequence

Sequence Name * Demo - Forms Builder and Workflow

Display Name * Demo - Forms Builder and Workflow

Description * Demo - Forms Builder and Workflow

Is Repeatable ☑

Confirmation Text * It worked!

Anonymous Mode ☐

Step 7: Create a workflow

a. Launch Workflow Designer.

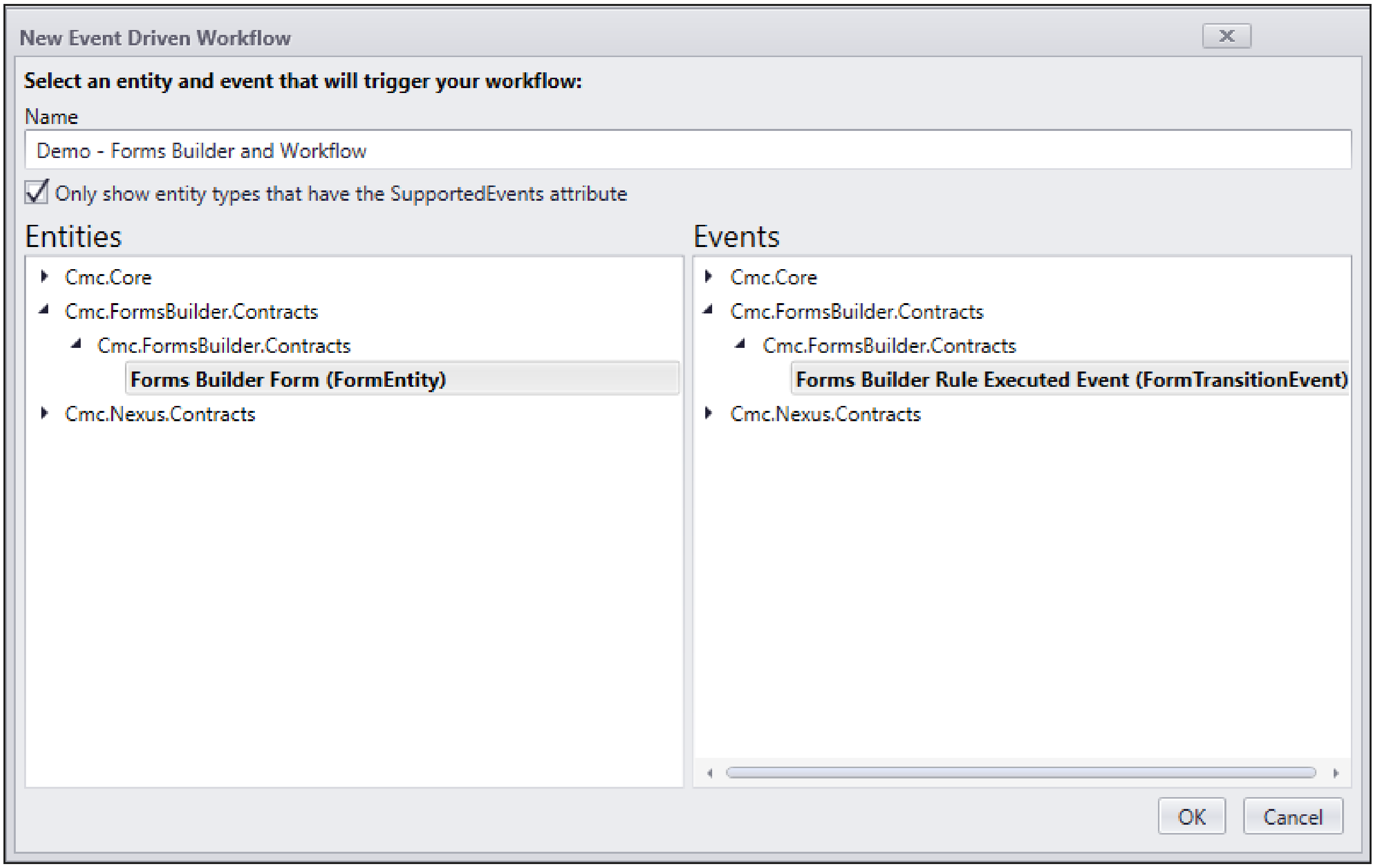
b. On the Home tab, click **New Event Workflow**.

c. In the Name field, type **Demo - Forms Builder and Workflow**.

d. In the Entities area, expand Cmc.FormsBuilder.Contracts and select **Forms Builder Form (FormEntity)**.

e. In the Events area, expand Cmc.FormsBuilder.Contracts and select **Forms Builder Rule Executed Event (FormTransitionEvent)**.

f. Click **OK**.


New Event Driven Workflow

Select an entity and event that will trigger your workflow:

Name

Demo - Forms Builder and Workflow

☑ Only show entity types that have the SupportedEvents attribute

Entities

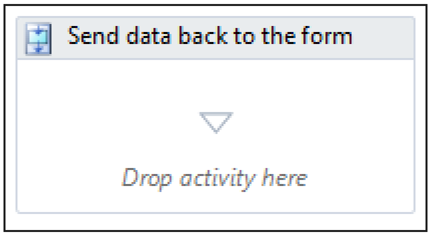
- Cmc.Core
- Cmc.FormsBuilder.Contracts
  - Cmc.FormsBuilder.Contracts
    - **Forms Builder Form (FormEntity)**
- Cmc.Nexus.Contracts

Events

- Cmc.Core
- Cmc.FormsBuilder.Contracts
  - Cmc.FormsBuilder.Contracts
    - **Forms Builder Rule Executed Event (FormTransitionEvent)**
- Cmc.Nexus.Contracts

Step 8: Rename the default Sequence

In the Properties pane, change the DisplayName to **Send data back to the form**.

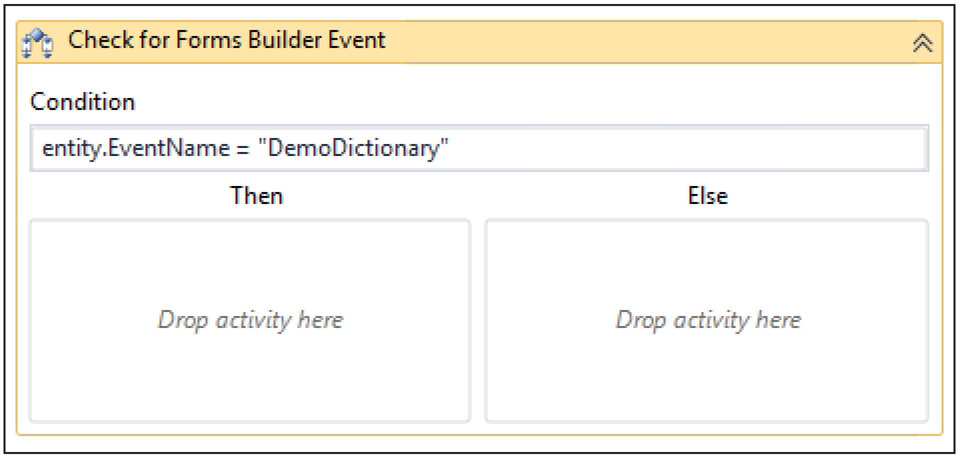

Send data back to the form

Step 9: Add an If activity to the workflow

a. In the Toolbox under Control Flow, find the **If** activity and drag it into the sequence.

b. In the Properties field, change the DisplayName to **Check for Forms Builder Event**.

c. In the Condition field, type: **entity.EventName = "DemoDictionary"**



Check for Forms Builder Event

Condition

entity.EventName = "DemoDictionary"

Step 10: Create variables

The first variable will hold the field name from the Forms Builder field we created.

a. In the Variables pane, create a variable named **FBField**.

b. In the Variable type field, select **String**.

c. The Scope field should be set to **Send data back to the form**.

d. In the Default field, type **"GetCurrentCourses"**. This is the name of the Forms Builder field exactly as we created it in step 1. Because it is a string, we must put it in quotes.

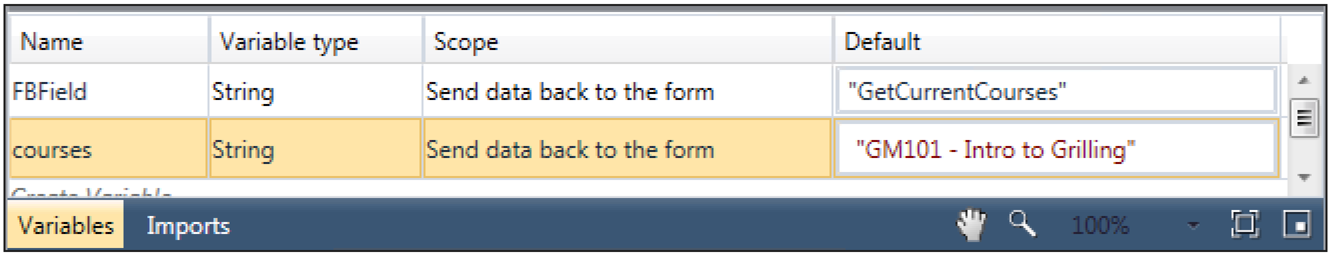
The next variable will hold the data we will send back to the Forms Builder form. Typically, we would query the information from the database, but for this purpose we will just populate it to simulate the data.

e. In the Variables pane, create a variable named **courses**.

f. In the Variable type field, select **String**.

g. The Scope field should be set to **Send data back to the form**.

h. In the Default field, type **"GM101 - Intro to Grilling"**.



Name	Variable type	Scope	Default
FBField	String	Send data back to the form	"GetCurrentCourses"
courses	String	Send data back to the form	"GM101 - Intro to Grilling"

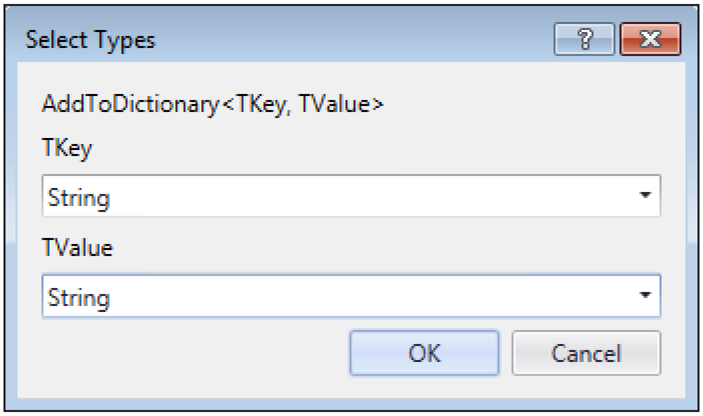
Step 11: Add to Dictionary

a. In the Toolbox under Cmc.Core.Workflow.Activities, find the **Add to Dictionary `2** activity and drag it into Then part of your If activity.

   This activity will pass information from the workflow back to a field inside of a Forms Builder form when a form transitions from one step to another.

b. When you drag the Add to Dictionary `2 activity into the Designer pane, you are prompted to set the data type. Forms Builder currently only supports strings.

   Select **String** in the TKey and TValue fields.



Select Types

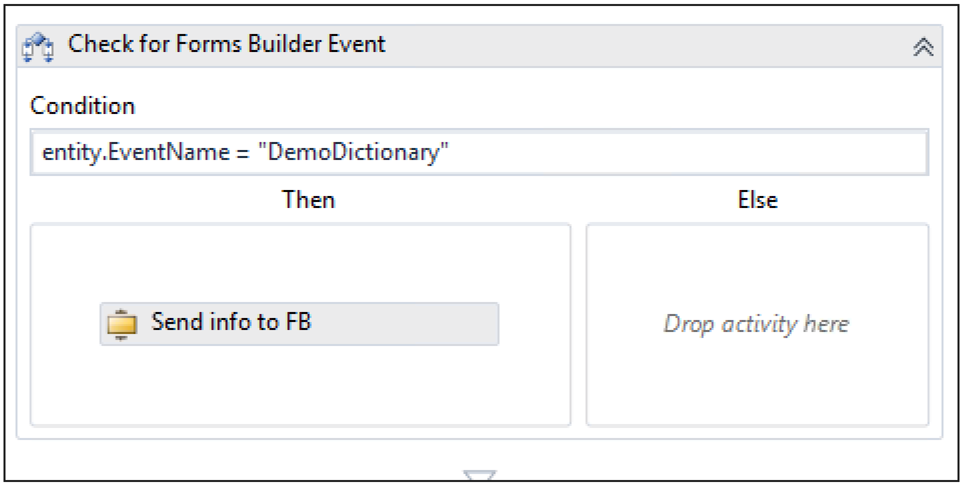
AddToDictionary<TKey, TValue>

TKey: String

TValue: String

c. In the Properties pane, set the Dictionary field to **args.DefaultFields**. This is the argument that is sending the data back to Forms Builder.

d. Change the DisplayName to **Send info to FB**.



Check for Forms Builder Event

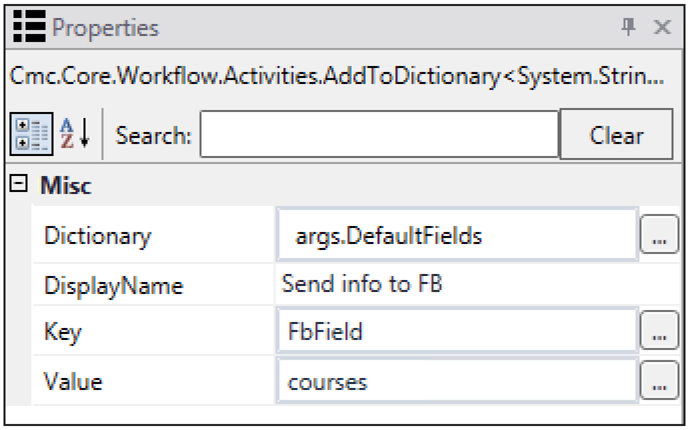
Condition: entity.EventName = "DemoDictionary"

Then: Send info to FB

Else: *Drop activity here*

e. In the Key field, specify the **FbField** variable. This is the field name of the custom field we created in step 1.

f. In the Value field, specify the **courses** variable. The value of this variable will show up on the Forms Builder form.



Properties

Cmc.Core.Workflow.Activities.AddToDictionary<System.Strin...

Search:

Misc

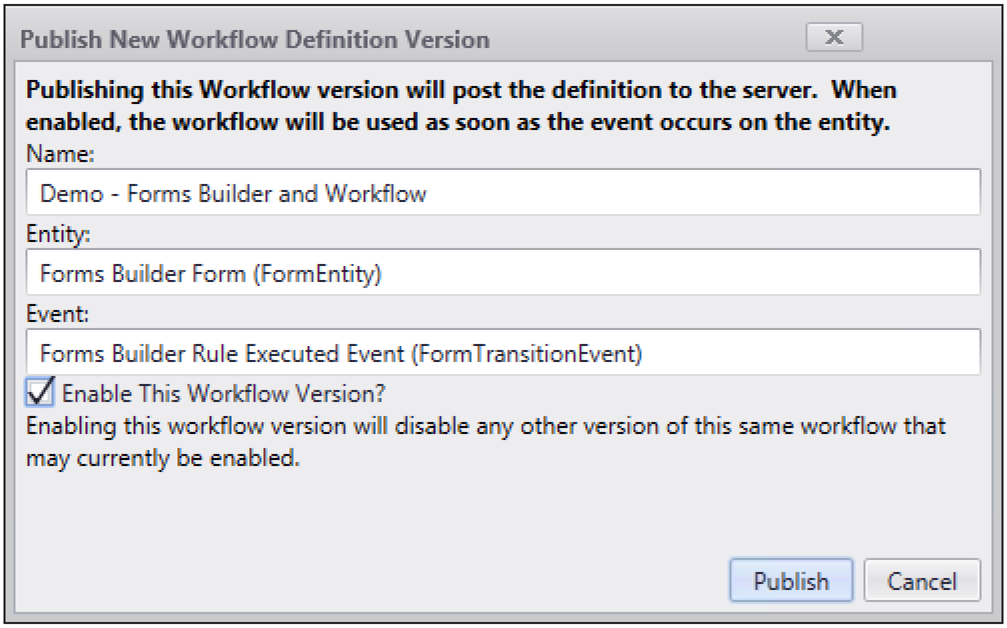
Dictionary	args.DefaultFields
DisplayName	Send info to FB
Key	FbField
Value	courses

Step 12: Publish the workflow

a. Check your workflow. Scroll through the workflow or use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.

b. On the Home tab, click **Publish**. The New Workflow Definition Version window is displayed.

c. Select **Enable This Workflow Version**



Publish New Workflow Definition Version

Publishing this Workflow version will post the definition to the server. When enabled, the workflow will be used as soon as the event occurs on the entity.

Name: Demo - Forms Builder and Workflow

Entity: Forms Builder Form (FormEntity)

Event: Forms Builder Rule Executed Event (FormTransitionEvent)

☑ Enable This Workflow Version?

Enabling this workflow version will disable any other version of this same workflow that may currently be enabled.

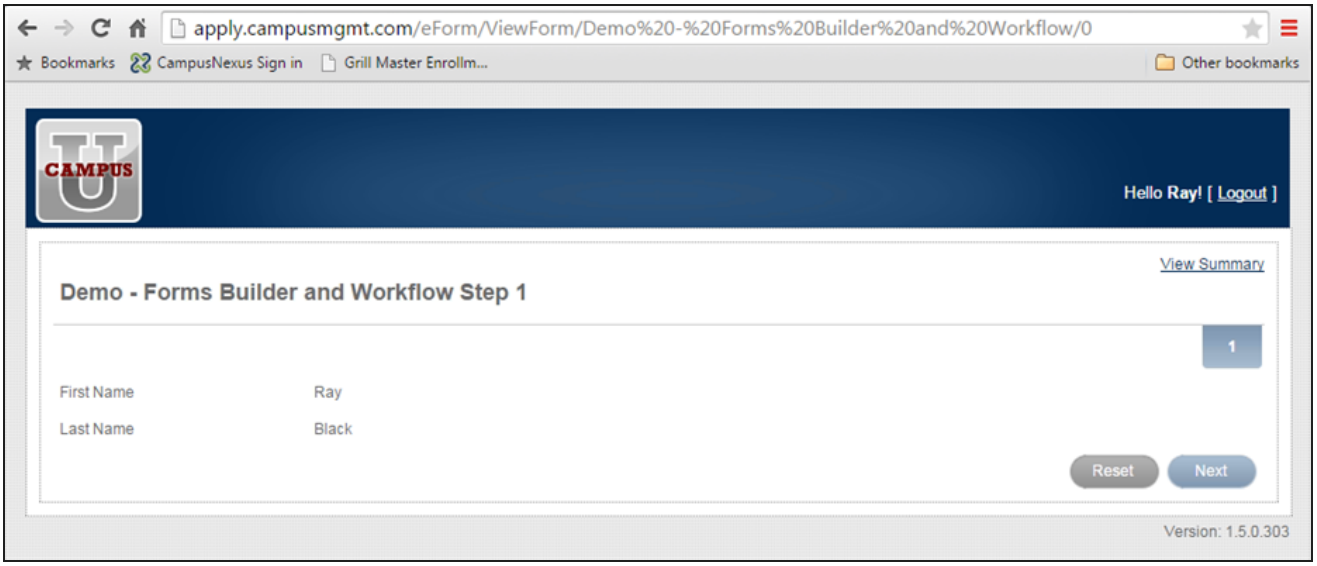
d. Click **Publish**, then **Cancel** to close the publisher window.

Step 13: Test the workflow

a. Forms Builder has a link to all of its published sequences, e.g., http://apply.campusmgmt.com/Home/PublishedSequences

On your Published Sequences page, select the Demo - Forms Builder and Workflow sequence and click **Open**.

b. Log in with the student Id. The Demo - Forms Builder and Workflow Step 1 form displays the First Name and Last Name field for the student.



apply.campusmgmt.com/eForm/ViewForm/Demo%20-%20Forms%20Builder%20and%20Workflow/0

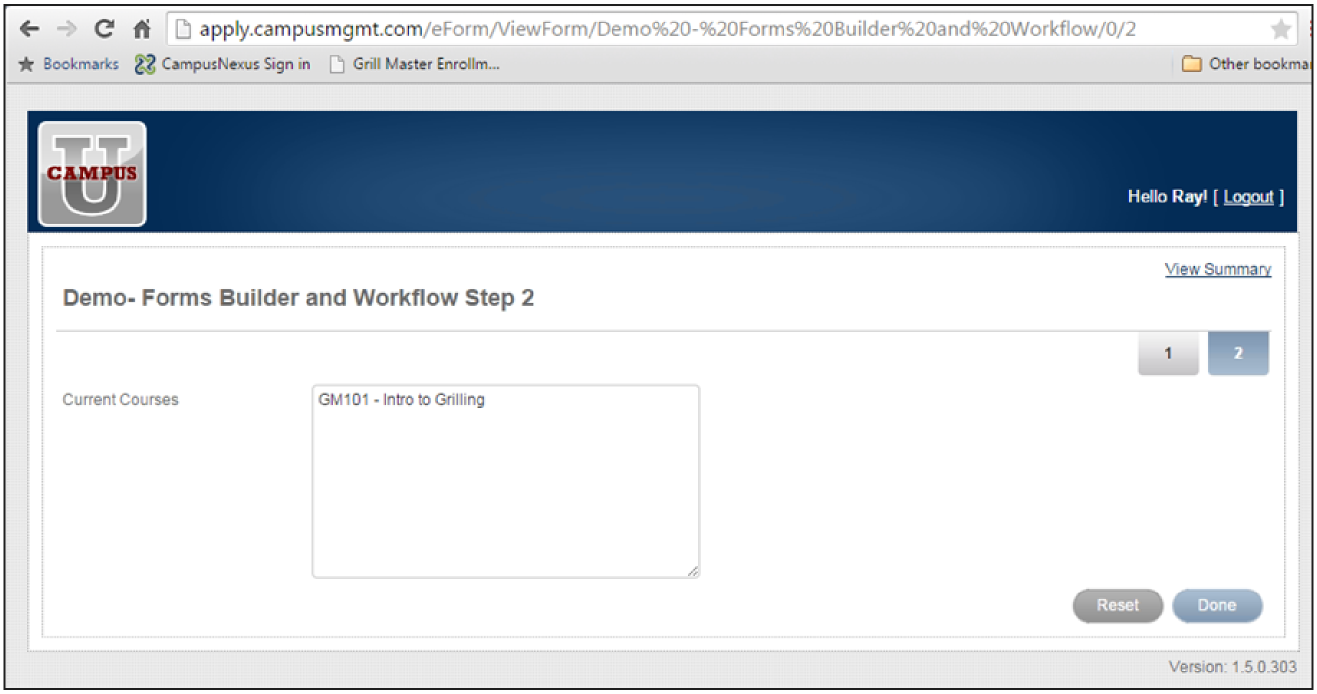
Demo - Forms Builder and Workflow Step 1

First Name Ray

Last Name Black

Version: 1.5.0.303

c. Click **Next**. The Demo - Forms Builder and Workflow Step 2 form is displayed. The Current Courses field shows the course the student is registered in.



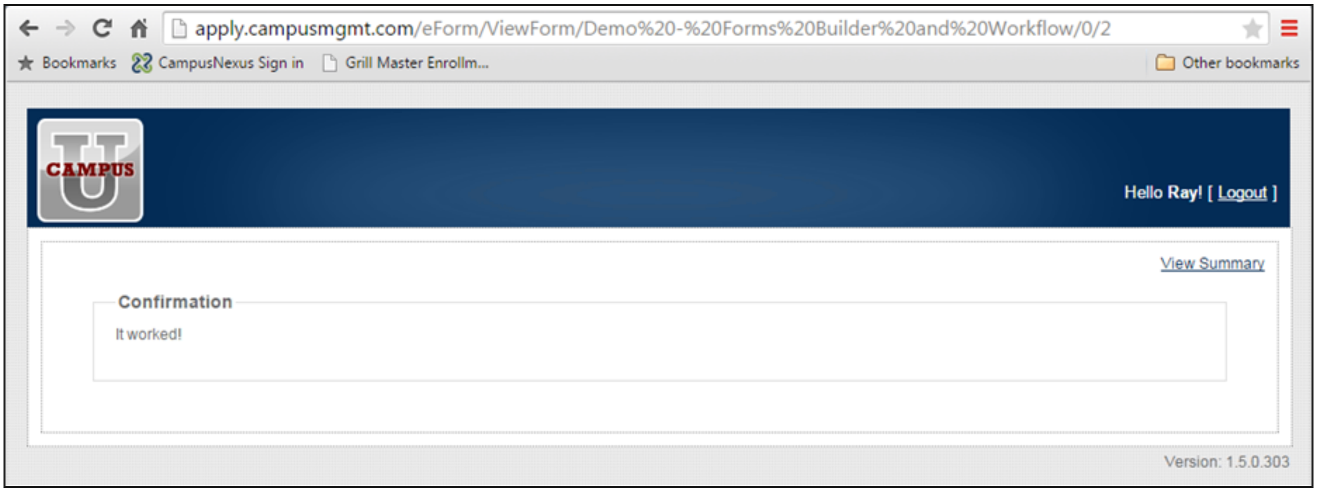
apply.campusmgmt.com/eForm/ViewForm/Demo%20-%20Forms%20Builder%20and%20Workflow/0/2

Demo- Forms Builder and Workflow Step 2

Current Courses GM101 - Intro to Grilling

Version: 1.5.0.303

d. Click **Done**. The Confirmation page is displayed.



Confirmation

It worked!

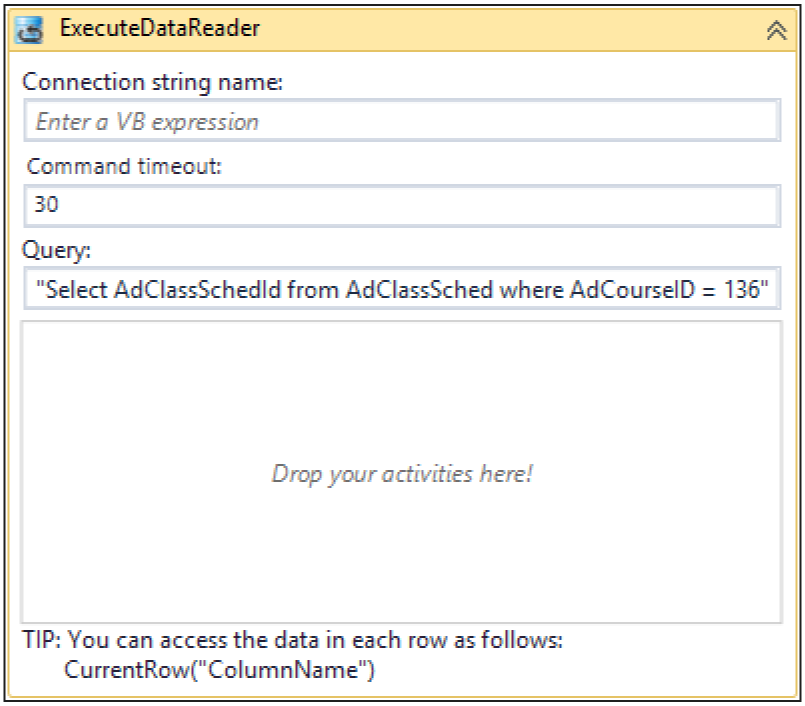
Register Students into a Course

This workflow finds students with a status of 'Future Start' and registers the students into an introductory course when their status changes.

1. Start the **Workflow** application from your desktop.
2. On the Home tab, click **New Event Workflow**.
3. In the Entities area:
    a. Click ▶ next to **Cmc.Nexus.Contracts**.
    b. Click ▶ next to **Cmc.Nexus.Sis.Academics**.
    c. Click **Student Enrollment Period (StudentEnrollmentPeriod)**.
4. In the Events area, click **Saved (SavedEvent)**.
5. Specify a **Name** for the workflow and click **OK**.
6. Drag a **LookupReferenceItem** activity it into the sequence.
    a. In the Reference Item Type field, select **Student Status**.
    b. In the Reference Item field, select **Future Start**.
    c. In the Variables pane, create a lookup type variable (**StudentStatus**) to contain the results of the lookup.
7. Drag an **If** activity into the sequence.
    a. In the Condition field, specify the following expression:

       entity.HasChanged(StudentEnrollmentPeriod.StudentStatusIdProperty) and entity.StudentStatusId.Value() = StudentStatus.Id
8. Drag an **ExecuteDataReader** activity into the Then branch of the If condition.
    a. In the Queryfield, specify the following string:

       "Select AdClassSchedId from AdClassSched where AdCourseID = 136"


ExecuteDataReader

Connection string name:

Enter a VB expression

Command timeout:

30

Query:

"Select AdClassSchedId from AdClassSched where AdCourseID = 136"

Drop your activities here!

TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

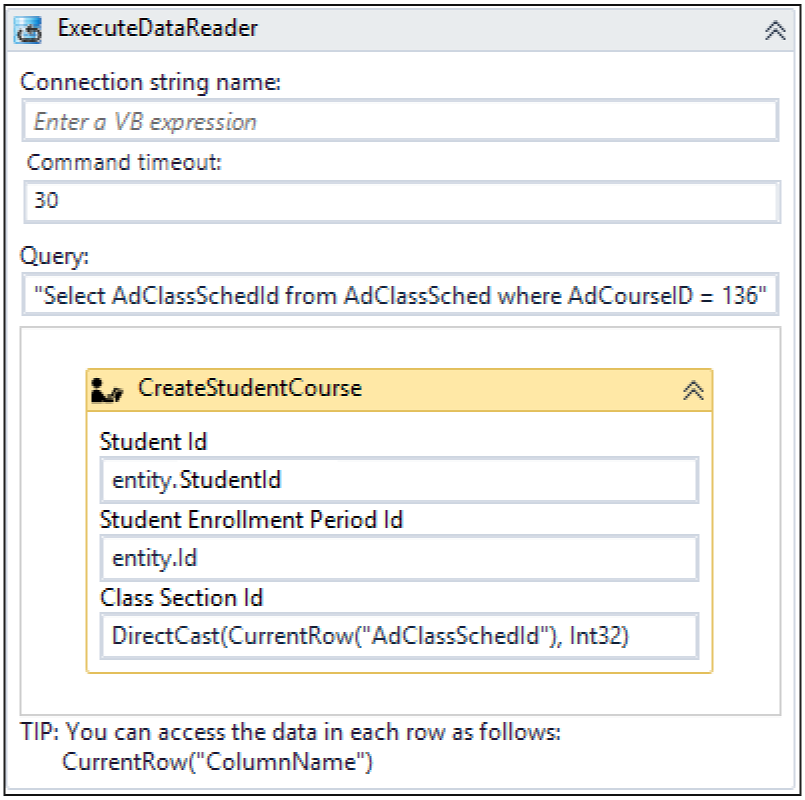
9. Drop a **CreateStudentCourse** activity into the ExecuteDataReader activity.

   a. In the Student Id field, specify **Student**.

   b. In the Student Enrollment Period Id field, specify **entity.Id**.

   c. In the Class Section Id field, specify **DirectCast(CurrentRow("AdClassSchedId"), Int32)**.

   d. In the Variables pane, create a variable to hold the Course object that you are creating and enter the variable name in the Properties pane for this activity.



ExecuteDataReader

Connection string name:

Enter a VB expression

Command timeout:

30

Query:

"Select AdClassSchedId from AdClassSched where AdCourseID = 136"

CreateStudentCourse

Student Id

entity.StudentId

Student Enrollment Period Id

entity.Id

Class Section Id

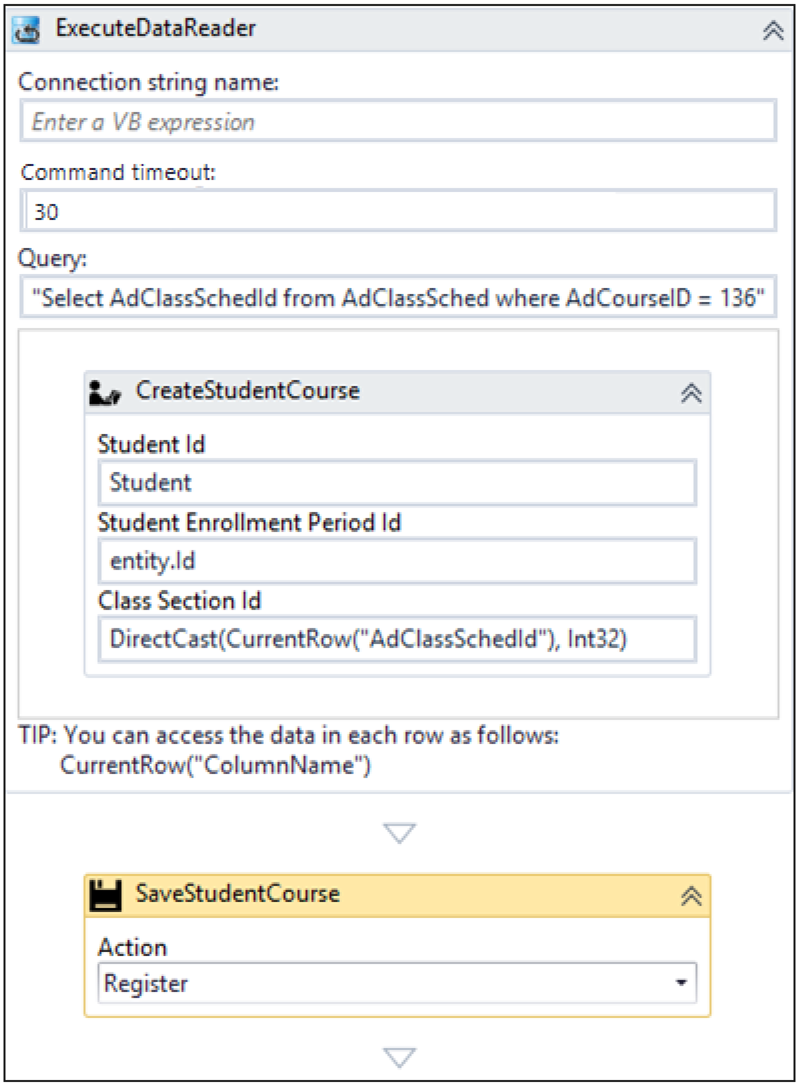
DirectCast(CurrentRow("AdClassSchedId"), Int32)

TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

10. Drag a **SaveStudentCourse** activity into the sequence below the ExecuteDataReader activity.

    a. In the Action field, select **Register**.

    b. Enter the variable that holds the Course object that you are creating in the Properties pane for this activity.



ExecuteDataReader

Connection string name:

Enter a VB expression

Command timeout:

30

Query:

"Select AdClassSchedId from AdClassSched where AdCourseID = 136"

CreateStudentCourse

Student Id

Student

Student Enrollment Period Id

entity.Id

Class Section Id

DirectCast(CurrentRow("AdClassSchedId"), Int32)

TIP: You can access the data in each row as follows:
CurrentRow("ColumnName")

SaveStudentCourse

Action

Register

11. Check your workflow. Scroll through the workflow or use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.
12. Click **Publish**. The New Workflow Definition Version window is displayed.
13. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.
14. Click **Save**, then **Cancel** to close the publisher window.

Transfer Students to Another Class Section

Scenario: Active students are enrolled and registered into a course in the current term. The course has multiple class sections in the current term. The course is in Scheduled status for the target students. The workflow is used to transfer students from one class section to another class section of the same course.

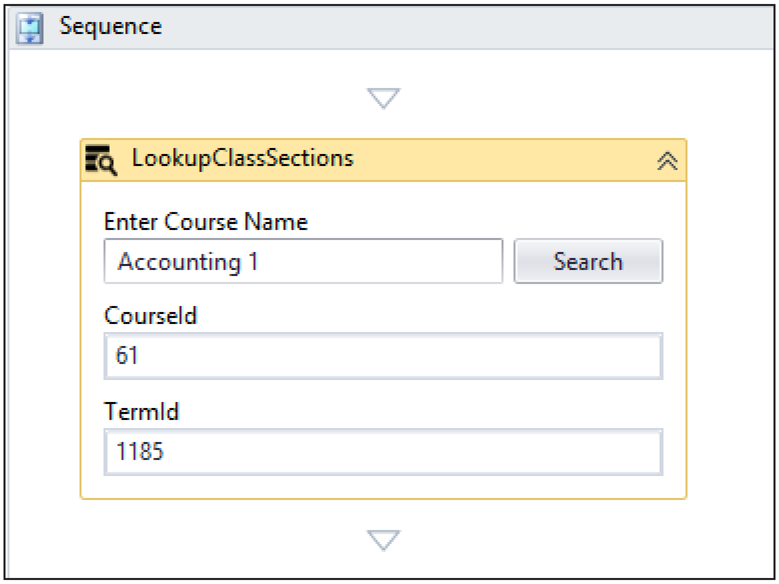
Similar workflows could be used to balance student populations across multiple class sections; or, if student groups are created, students could be transferred into a class section based on specific student group criteria.

1. Start the **Workflow** application from your desktop.
2. On the Home tab, click **New Event Workflow**.
3. In the Entities area:
   1. Click ▸ next to **Cmc.Nexus.Contracts**.
   2. Click ▸ next to **Cmc.Nexus**.
   3. Click **Group Membership (GroupMembership)**.
4. In the Events area, click **Saved (SavedEvent)**.
5. Specify a **Name** for the workflow and click **OK**.
6. Drop a **LookupClassSections** activity into the sequence.
   1. Click the **Search** button to find and select the course into which the students are transferred.
   2. Create a variable for the array of class section values.

      Name the variable, e.g., **ClassSects**, and select the Variable type of **ClassSection[]**.
   3. Specify appropriate values for the **CourseId** and **TermId**.

      Note: Use an SQL query to determine the CourseId and TermId values for your database environment, e.g.,

      ```
      select * from AdEnrollSched where AdCourseID = "value from workflow" and systudentid = "current student id"
      ```



Sequence

LookupClassSections

Enter Course Name: Accounting 1

CourseId: 61

TermId: 1185

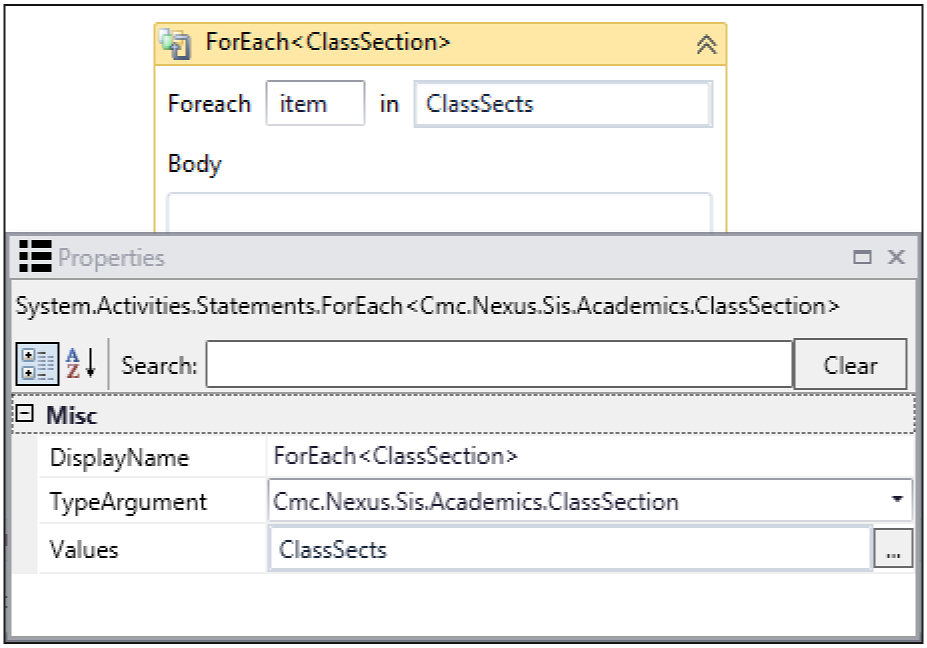
7. Drop a **For Each<>** activity into the sequence.

   a. In the TypeArgument field, browse for **Cmc.Nexus.Sis.Academics.ClassSection**.

      The DisplayName field changes to ForEach<ClassSection>.

   b. In the Values field, enter the **ClassSects** variable created in the previous step.

The ForEach<ClassSection> activity steps through the array of class sections and returns the Ids for each item.



ForEach<ClassSection>

Foreach item in ClassSects

Body

System.Activities.Statements.ForEach<Cmc.Nexus.Sis.Academics.ClassSection>

Misc

DisplayName	ForEach<ClassSection>
TypeArgument	Cmc.Nexus.Sis.Academics.ClassSection
Values	ClassSects

8. Drop a **Sequence** into the Body field of the ForEach<ClassSection> activity.

9. Drop an **If** activity into the sequence added in the previous step.

a. In the Condition field, specify the following expression:

   item.SectionCodeEquals("ACC101")

   where "ACC101" (case sensitive) is the name of the class section into which you want to transfer students.

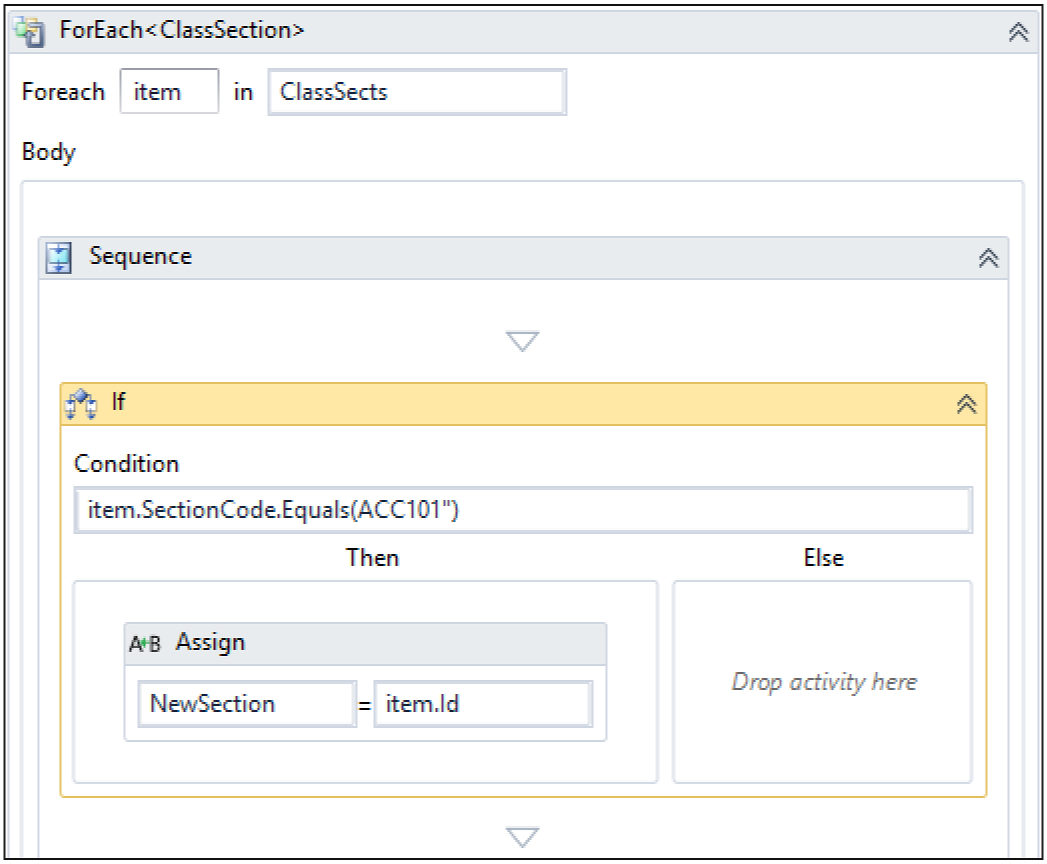
10. Create a variable for the new class section values.

    Name the variable, e.g., **NewSection**, and select the Variable type of **Int32**.

11. Drop an **A+B Assign** activity into the Then branch of the If condition.

    a. In the To field of the Assign activity, specify the name of the variable created in the previous step (NewSection).

    b. Assign the value **item.ID** to the NewSection variable.



ForEach<ClassSection>

Foreach item in ClassSects

Body

Sequence

If

Condition

item.SectionCode.Equals(ACC101")

Then

Else

A+B Assign

NewSection = item.Id

12. Drop a **SaveStudentCourse** activity into the sequence below the If activity.

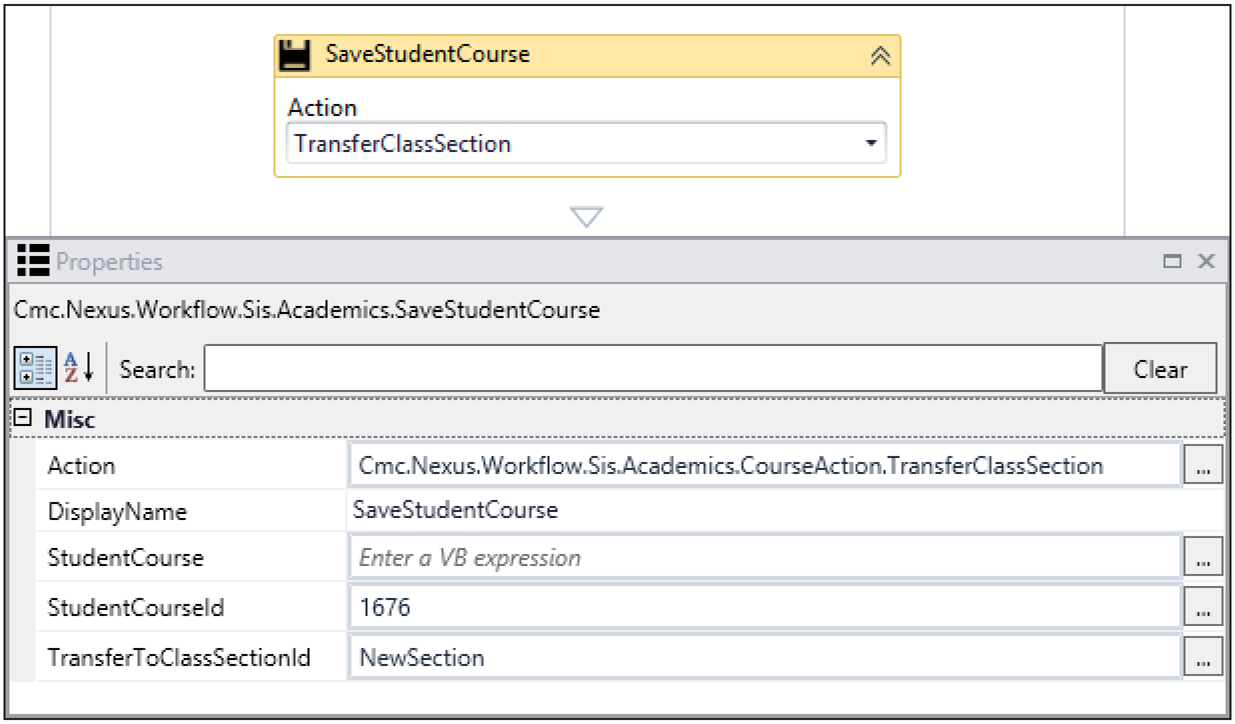
    a. In the Action field, select **TransferClassSection**.

    b. In the StudentCourseId field, specify the value of the current class.

Note: Use an SQL query to determine the StudentCourseId value for your database environment, e.g.,

`select adenrollschedid from AdEnrollSched where AdCourseID =` "value from workflow" `and systudentid =` "current student id"

c. In the TransferToClassSectionId field, specify the **NewSection** variable.



Action	Cmc.Nexus.Workflow.Sis.Academics.CourseAction.TransferClassSection
DisplayName	SaveStudentCourse
StudentCourse	*Enter a VB expression*
StudentCourseId	1676
TransferToClassSectionId	NewSection

13. Check your workflow. Scroll through the workflow or use the fit to screen button located at the bottom of the Designer pane to see the whole workflow based on your screen resolution.

14. Click **Publish**. The New Workflow Definition Version window is displayed.

15. If you want the workflow to be run as soon as the event occurs on the entity, select **Enable This Workflow Version**, otherwise leave the check box cleared.

16. Click **Save**, then **Cancel** to close the publisher window.

Resources

This section contains reference material that may assist you when designing and testing workflows.

Related Help Systems and APIs

https://help.campusmanagement.com/Content/Home.htm

https://www.mycampusinsight.com/Documentation-Center/Help/Help_Home/Content/helphome.htm (logon required). The Object Library for Anthology Student is available under APIs > Anthology Student Object Library.

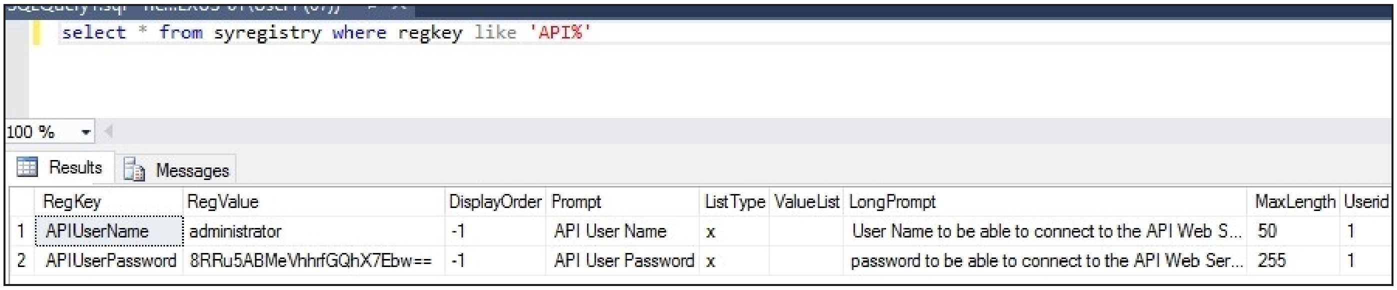
API Errors with SyRegistry Authentication

Note: Anthology Student 22.0 introduces an alternate method for the authentication of CampusLink API calls. The new authentication method does not use the SyRegistry table. For details see Authentication for CampusLink API Calls.

API Password

If the below error is received in Workflow Composer and/or in the logs, the API Password that is in the SyRegistry table is not correct. To sync the password, log in to the Portal Admin Console and update the password for the API user.

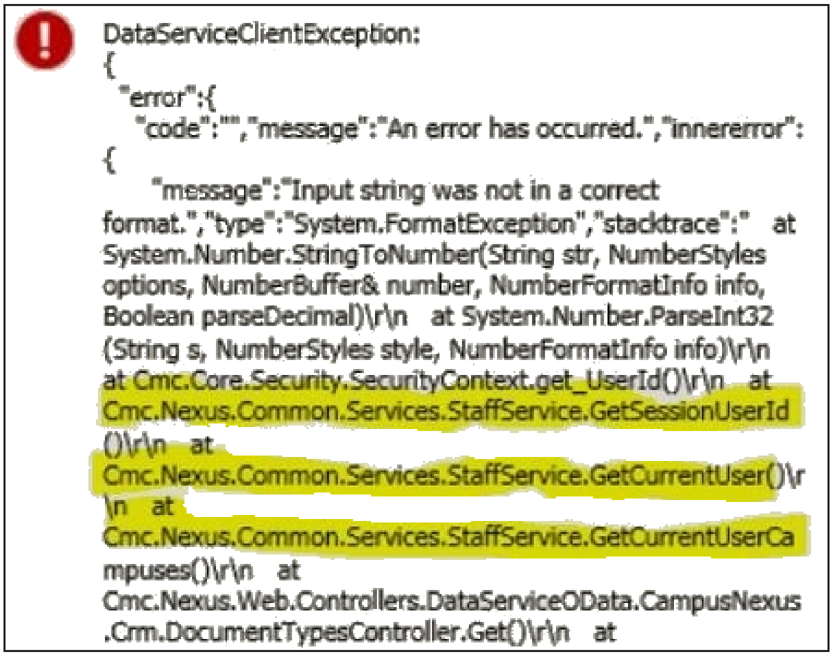
SyRegistry query:



```
select * from syregistry where regkey like 'API%'
```

	RegKey	RegValue	DisplayOrder	Prompt	ListType	ValueList	LongPrompt	MaxLength	Userid
1	APIUserName	administrator	-1	API User Name	x		User Name to be able to connect to the API Web S...	50	1
2	APIUserPassword	8RRu5ABMeVhhrfGQhX7Ebw==	-1	API User Password	x		password to be able to connect to the API Web Ser...	255	1

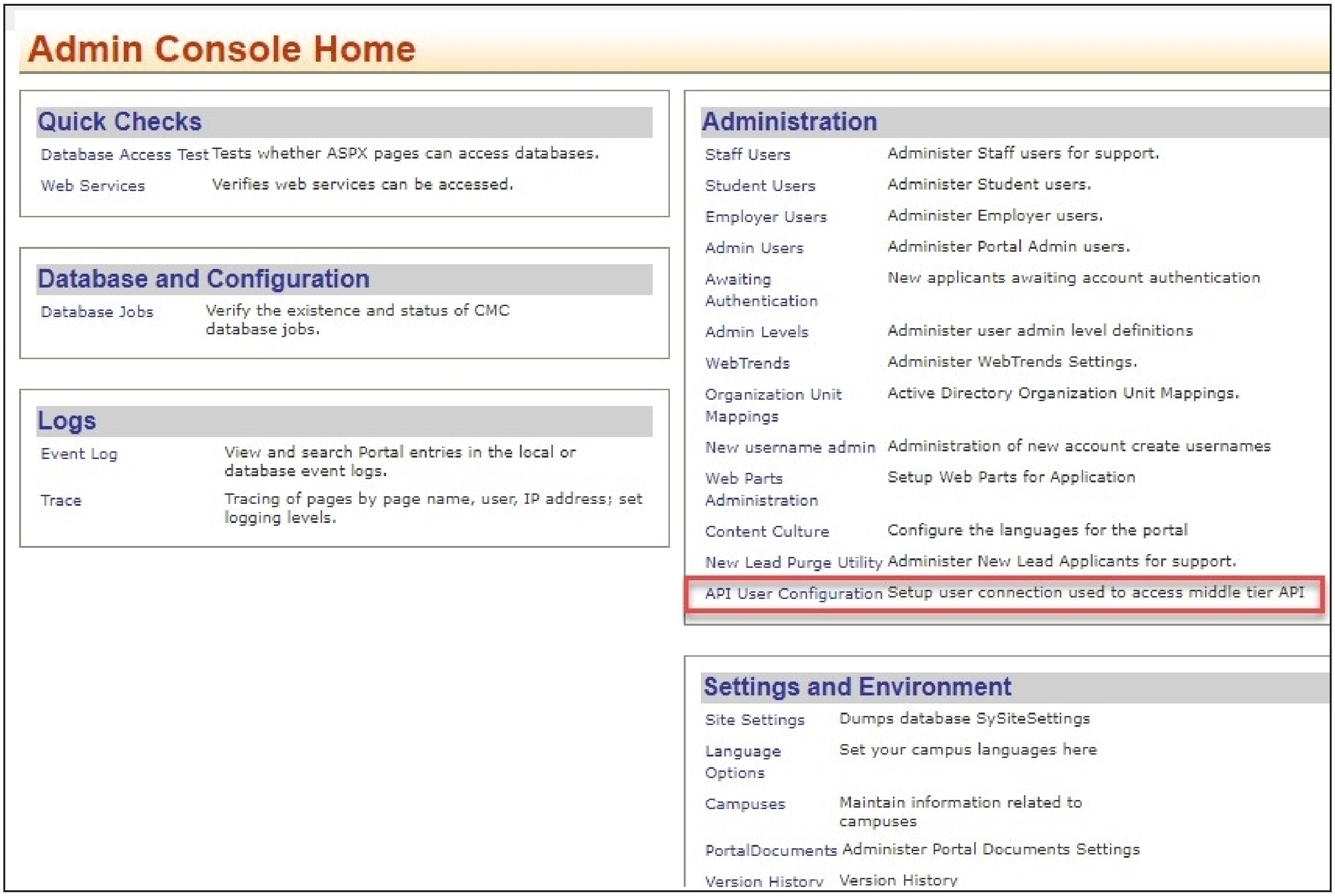
Workflow and Log Error:



```
DataServiceClientException:
{
  "error":{
    "code":"","message":"An error has occurred.","innererror":
{
      "message":"Input string was not in a correct
format.","type":"System.FormatException","stacktrace":"   at
System.Number.StringToNumber(String str, NumberStyles
options, NumberBuffer& number, NumberFormatInfo info,
Boolean parseDecimal)\r\n   at System.Number.ParseInt32
(String s, NumberStyles style, NumberFormatInfo info)\r\n
at Cmc.Core.Security.SecurityContext.get_UserId()\r\n   at
Cmc.Nexus.Common.Services.StaffService.GetSessionUserId
()\r\n   at
Cmc.Nexus.Common.Services.StaffService.GetCurrentUser()\r
\n   at
Cmc.Nexus.Common.Services.StaffService.GetCurrentUserCa
mpuses()\r\n   at
Cmc.Nexus.Web.Controllers.DataServiceOData.CampusNexus
.Crm.DocumentTypesController.Get()\r\n   at
```

```
Cmc.Core.Eventing.EventHandlerException: An exception was thrown within an event handler. --->
System.NullReferenceException: Object reference not set to an instance of an object.
 at Cmc.Nexus.Common.Services.StaffService.GetApiUserId()
 at Cmc.Nexus.Common.Services.StaffService.GetSessionUserId()
 at Cmc.Nexus.Common.Services.StaffService.GetCurrentUser()
 at Cmc.Nexus.Common.EventHandlers.CommonEventHandlers.SetAuditableFields(Object entity, Boolean
isNewEntity)
```

Portal Admin Console:



Admin Console Home

Quick Checks

Database Access Test	Tests whether ASPX pages can access databases.
Web Services	Verifies web services can be accessed.

Database and Configuration

Database Jobs	Verify the existence and status of CMC database jobs.

Logs

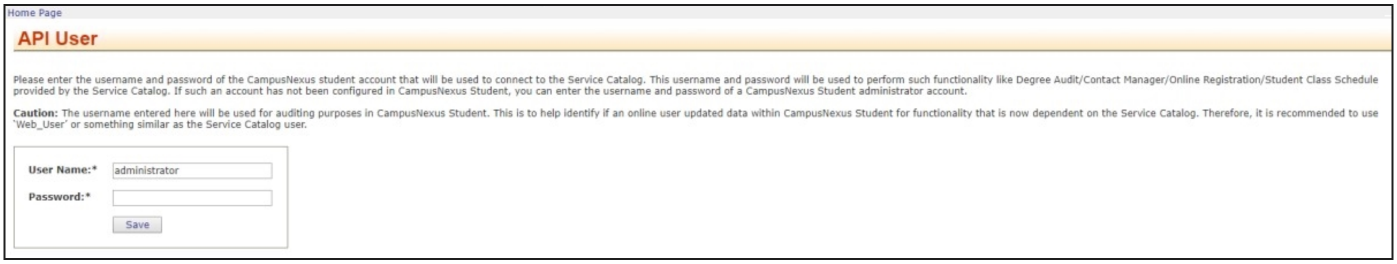
Event Log	View and search Portal entries in the local or database event logs.
Trace	Tracing of pages by page name, user, IP address; set logging levels.

Administration

Staff Users	Administer Staff users for support.
Student Users	Administer Student users.
Employer Users	Administer Employer users.
Admin Users	Administer Portal Admin users.
Awaiting Authentication	New applicants awaiting account authentication
Admin Levels	Administer user admin level definitions
WebTrends	Administer WebTrends Settings.
Organization Unit Mappings	Active Directory Organization Unit Mappings.
New username admin	Administration of new account create usernames
Web Parts Administration	Setup Web Parts for Application
Content Culture	Configure the languages for the portal
New Lead Purge Utility	Administer New Lead Applicants for support.
API User Configuration	Setup user connection used to access middle tier API

Settings and Environment

Site Settings	Dumps database SySiteSettings
Language Options	Set your campus languages here
Campuses	Maintain information related to campuses
PortalDocuments	Administer Portal Documents Settings
Version History	Version History



API User

Please enter the username and password of the CampusNexus student account that will be used to connect to the Service Catalog. This username and password will be used to perform such functionality like Degree Audit/Contact Manager/Online Registration/Student Class Schedule provided by the Service Catalog. If such an account has not been configured in CampusNexus Student, you can enter the username and password of a CampusNexus Student administrator account.

Caution: The username entered here will be used for auditing purposes in CampusNexus Student. This is to help identify if an online user updated data within CampusNexus Student for functionality that is now dependent on the Service Catalog. Therefore, it is recommended to use 'Web_User' or something similar as the Service Catalog user.

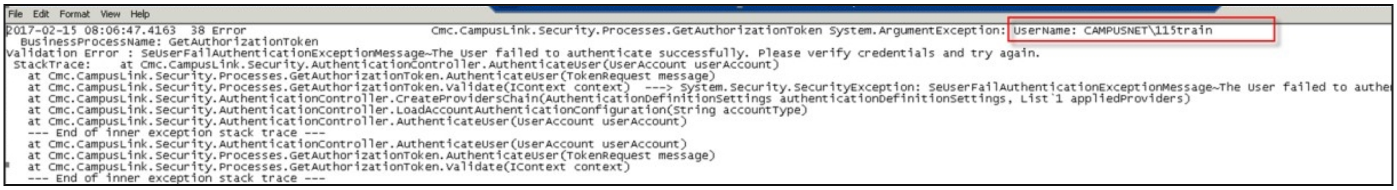
User Name:* administrator

Password:*

API User Permissions

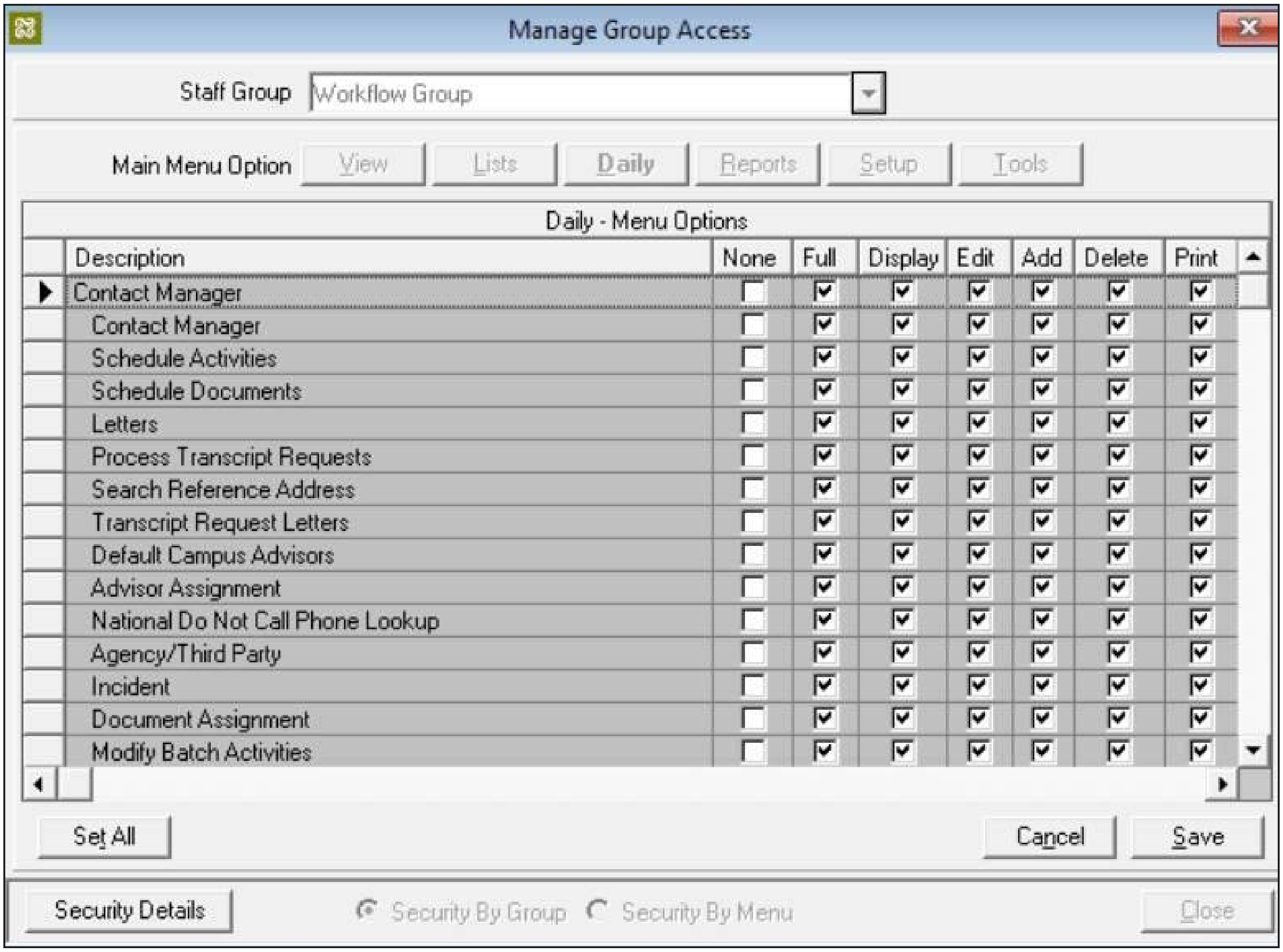
The API User specified in the SyRegistry table has to have permissions to execute the CampusLink APIs. This user must exist in Anthology Student and be part of a group other than the Administrator group that has full permissions to the Daily menu. This user also needs to be assigned the proper Activity Security and Document Security policies.

Possible Error Received in Log File if Permissions are not Correct:



```
2017-02-15 08:06:47.4163  38 Error                 Cmc.CampusLink.Security.Processes.GetAuthorizationToken System.ArgumentException: UserName: CAMPUSNET\115train
  BusinessProcessName: GetAuthorizationToken
Validation Error : SeUserFailAuthenticationExceptionMessage~The User failed to authenticate successfully. Please verify credentials and try again.
 StackTrace:    at Cmc.CampusLink.Security.AuthenticationController.AuthenticateUser(UserAccount userAccount)
   at Cmc.CampusLink.Security.Processes.GetAuthorizationToken.AuthenticateUser(TokenRequest message)
   at Cmc.CampusLink.Security.Processes.GetAuthorizationToken.Validate(IContext context)  ---> System.Security.SecurityException: SeUserFailAuthenticationExceptionMessage~The User failed to authe
   at Cmc.CampusLink.Security.AuthenticationController.CreateProvidersChain(AuthenticationDefinitionSettings authenticationDefinitionSettings, List`1 appliedProviders)
   at Cmc.CampusLink.Security.AuthenticationController.LoadAccountAuthenticationConfiguration(String accountType)
   at Cmc.CampusLink.Security.AuthenticationController.AuthenticateUser(UserAccount userAccount)
   --- End of inner exception stack trace ---
   at Cmc.CampusLink.Security.AuthenticationController.AuthenticateUser(UserAccount userAccount)
   at Cmc.CampusLink.Security.Processes.GetAuthorizationToken.AuthenticateUser(TokenRequest message)
   at Cmc.CampusLink.Security.Processes.GetAuthorizationToken.Validate(IContext context)
   --- End of inner exception stack trace ---
```

Anthology Student Configuration:

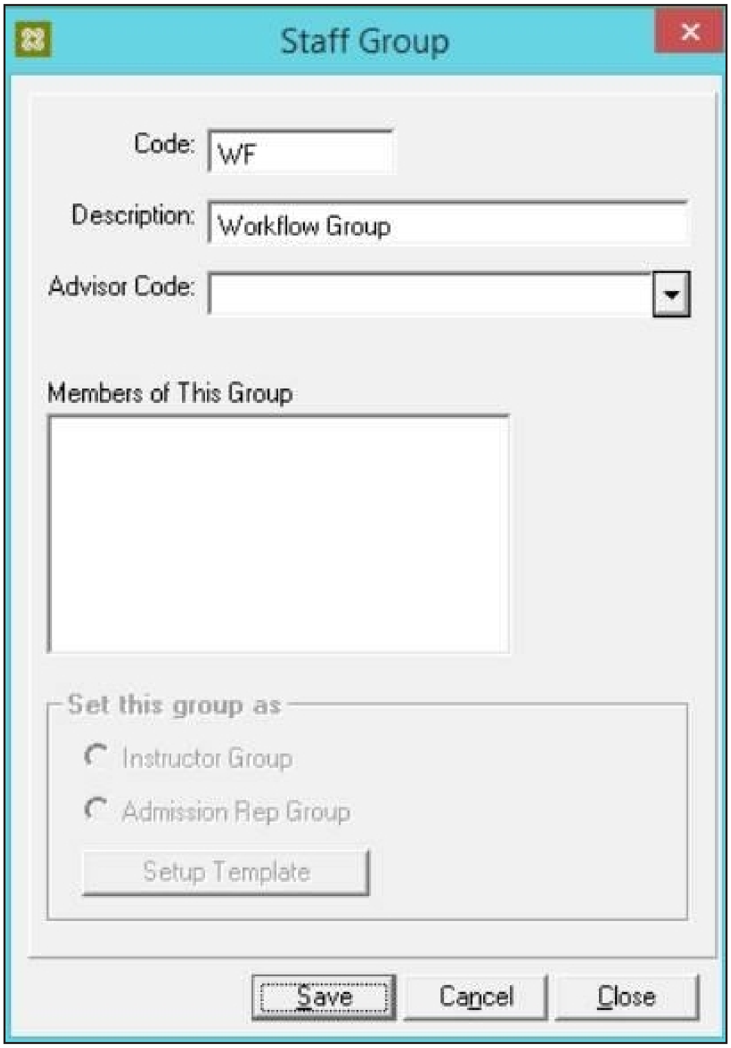


Staff Group: Workflow Group

● Security By Group ○ Security By Menu

Daily - Menu Options

Description	None	Full	Display	Edit	Add	Delete	Print
Contact Manager	□	☑	☑	☑	☑	☑	☑
Contact Manager	□	☑	☑	☑	☑	☑	☑
Schedule Activities	□	☑	☑	☑	☑	☑	☑
Schedule Documents	□	☑	☑	☑	☑	☑	☑
Letters	□	☑	☑	☑	☑	☑	☑
Process Transcript Requests	□	☑	☑	☑	☑	☑	☑
Search Reference Address	□	☑	☑	☑	☑	☑	☑
Transcript Request Letters	□	☑	☑	☑	☑	☑	☑
Default Campus Advisors	□	☑	☑	☑	☑	☑	☑
Advisor Assignment	□	☑	☑	☑	☑	☑	☑
National Do Not Call Phone Lookup	□	☑	☑	☑	☑	☑	☑
Agency/Third Party	□	☑	☑	☑	☑	☑	☑
Incident	□	☑	☑	☑	☑	☑	☑
Document Assignment	□	☑	☑	☑	☑	☑	☑
Modify Batch Activities	□	☑	☑	☑	☑	☑	☑



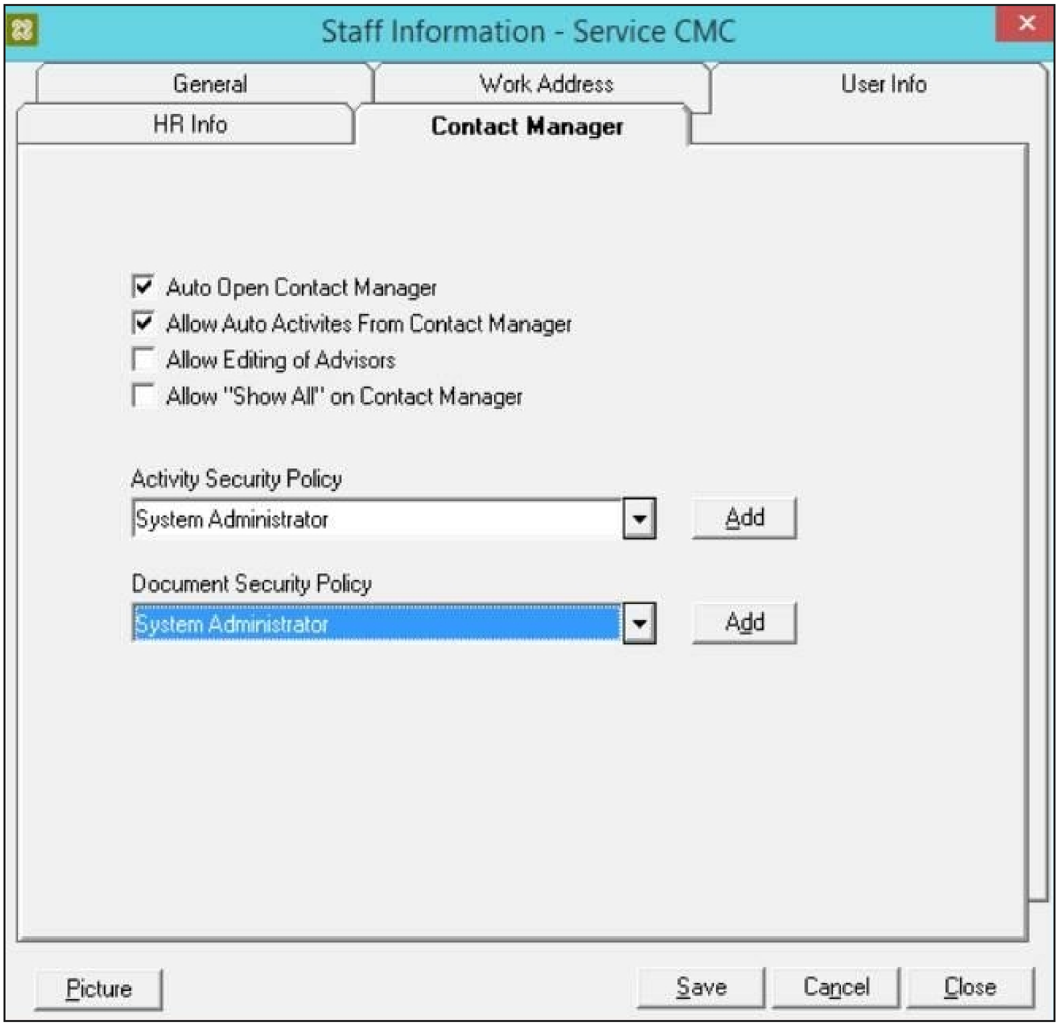
Code: WF

Description: Workflow Group

Advisor Code:

○ Instructor Group

○ Admission Rep Group



☑ Auto Open Contact Manager
☑ Allow Auto Activites From Contact Manager
□ Allow Editing of Advisors
□ Allow "Show All" on Contact Manager

Activity Security Policy: System Administrator

Document Security Policy: System Administrator

API Key – Access Denied Error

If the API keys are not set up correctly, an "Access denied" error will be seen in the Renderer log, for example, when a Forms Builder workflow calls the Anthology Student activity.

Solution: Ensure that the API keys across all products match.

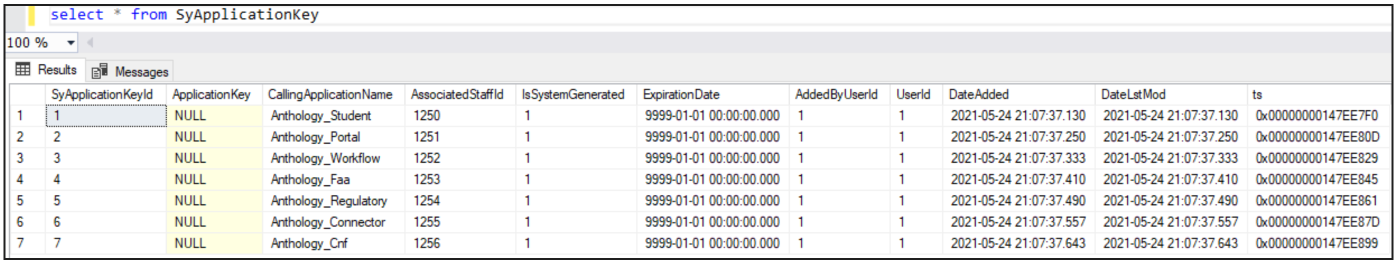
```
<appSettings>
    <add key="ConfigureCampusNexusWcfProxy" value="true" />
    <add key="ConfigureCVueNexusWcfProxy" value="true" />
    <!-- Following will be populated when Crm is enabled for Forms Builder -->
    <add key="CmcNexusCrmWebUrl" value="http://<server:port>/" />
    <add key="PaymentProvider" value="pilot-payflowpro.paypal.com" />
    <add key="AuxiliaryServiceBaseUrl" value="" />
    <!-- Following should be set to true only in Azure environments where the Auxiliary service
is installed and required. -->
    <add key="UseRemotePdfConversionService" value="false" />
    <!-- Following sets a time before conversion to PDF starts. Default 500, increase if blank
documents on a slow server. -->
    <add key="ViewCreatorDefaultStartConversionTimerInMilliseconds" value="" />
    <add key="ApiKey" value="<Your API key value>" />
  </appSettings>
```

Authentication for CampusLink API Calls

Many calls from Anthology Student, Portal, Workflow activities, and other integrated applications rely upon a valid staff account (“APIuser”) to make CampusLink API calls. This account needs to be unique (i.e., not used for anything else). The user name and password for the account are stored in the database (SyRegistry table). The account must also exist in Active Directory (AD) or Azure Active Directory (AAD). The account details in the database and in AD/AAD need to be in sync. This can create maintenance and security issues (e.g., multifactor authentication (MFA) needs to be disabled). Therefore, the existing authentication mechanism using SyRegistry keys with user name and password will eventually be retired. For the time being, the existing logic based on a user name and password continues to work to satisfy backward compatibility requirements for integrated products.

The July 2021 releases of Anthology products introduce an alternate authentication mechanism that relies on symmetric keys. A new SyApplicationKey table stores the encrypted keys and names of the calling applications. The keys are decrypted before they are passed to the CampusLink Authentication Service.

At the time of Anthology Student 22.0 installation, a script inserts 1 record per calling application (with ApplicationKey value = NULL) into the SyApplicationKey table (see image below). The script also inserts staff users with the necessary permissions for all Anthology products that use key-based authentication and updates the corresponding AssociatedStaffId. The AssociatedStaffId is the identity that will be used for CampusLink API calls. The script makes the staff users part of the necessary staff groups to apply the required permissions. Institutions no longer need to manually create staff users for CampusLink API calls.



```
select * from SyApplicationKey
```

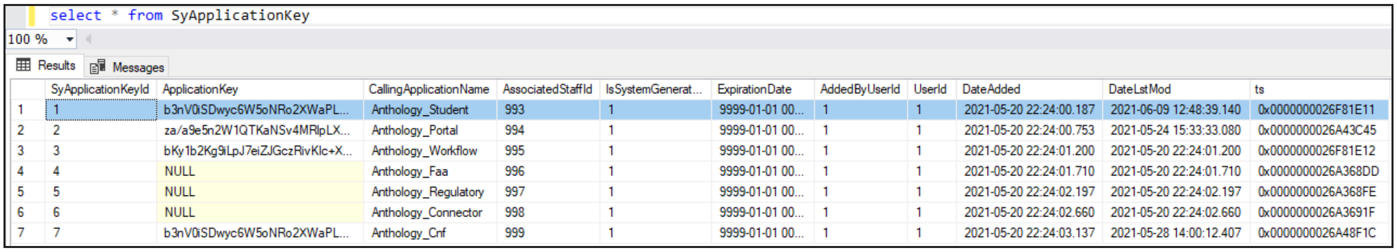
	SyApplicationKeyId	ApplicationKey	CallingApplicationName	AssociatedStaffId	IsSystemGenerated	ExpirationDate	AddedByUserId	UserId	DateAdded	DateLstMod	ts
1	1	NULL	Anthology_Student	1250	1	9999-01-01 00:00:00.000	1	1	2021-05-24 21:07:37.130	2021-05-24 21:07:37.130	0x00000000147EE7F0
2	2	NULL	Anthology_Portal	1251	1	9999-01-01 00:00:00.000	1	1	2021-05-24 21:07:37.250	2021-05-24 21:07:37.250	0x00000000147EE80D
3	3	NULL	Anthology_Workflow	1252	1	9999-01-01 00:00:00.000	1	1	2021-05-24 21:07:37.333	2021-05-24 21:07:37.333	0x00000000147EE829
4	4	NULL	Anthology_Faa	1253	1	9999-01-01 00:00:00.000	1	1	2021-05-24 21:07:37.410	2021-05-24 21:07:37.410	0x00000000147EE845
5	5	NULL	Anthology_Regulatory	1254	1	9999-01-01 00:00:00.000	1	1	2021-05-24 21:07:37.490	2021-05-24 21:07:37.490	0x00000000147EE861
6	6	NULL	Anthology_Connector	1255	1	9999-01-01 00:00:00.000	1	1	2021-05-24 21:07:37.557	2021-05-24 21:07:37.557	0x00000000147EE87D
7	7	NULL	Anthology_Cnf	1256	1	9999-01-01 00:00:00.000	1	1	2021-05-24 21:07:37.643	2021-05-24 21:07:37.643	0x00000000147EE899

On the first CampusLink API call, the key-based authentication logic:

- Generates a key for the product identified by the CallingApplicationName,
- Encrypts and saves the key to the SyApplicationKey table, and
- Retrieves the decrypted key.

On subsequent CampusLink API calls, the logic retrieves the existing key and passes it on to CampusLink.

Note: Since Workflow always executes in the context of Anthology Student, workflow uses Anthology_Student as the CallingApplicationName.



```
select * from SyApplicationKey
```

	SyApplicationKeyId	ApplicationKey	CallingApplicationName	AssociatedStaffId	IsSystemGenerat...	ExpirationDate	AddedByUserId	UserId	DateAdded	DateLstMod	ts
1	1	b3nV0iSDwyc6W5oNRo2XWaPL...	Anthology_Student	993	1	9999-01-01 00...	1	1	2021-05-20 22:24:00.187	2021-06-09 12:48:39.140	0x000000026F81E11
2	2	za/a9e5n2W1QTKaNSv4MRlpLX...	Anthology_Portal	994	1	9999-01-01 00...	1	1	2021-05-20 22:24:00.753	2021-05-24 15:33:33.080	0x000000026A43C45
3	3	bKy1b2Kg9iLpJ7eiZJGczRivKlc+X...	Anthology_Workflow	995	1	9999-01-01 00...	1	1	2021-05-20 22:24:01.200	2021-05-20 22:24:01.200	0x000000026F81E12
4	4	NULL	Anthology_Faa	996	1	9999-01-01 00...	1	1	2021-05-20 22:24:01.710	2021-05-20 22:24:01.710	0x000000026A368DD
5	5	NULL	Anthology_Regulatory	997	1	9999-01-01 00...	1	1	2021-05-20 22:24:02.197	2021-05-20 22:24:02.197	0x000000026A368FE
6	6	NULL	Anthology_Connector	998	1	9999-01-01 00...	1	1	2021-05-20 22:24:02.660	2021-05-20 22:24:02.660	0x000000026A3691F
7	7	b3nV0iSDwyc6W5oNRo2XWaPL...	Anthology_Cnf	999	1	9999-01-01 00...	1	1	2021-05-20 22:24:03.137	2021-05-28 14:00:12.407	0x000000026A48F1C

CampusLink Authentication Service Updates

To support the key-based security, the CampusLink Authentication Service is modified by adding the following values to the in-message of the GetAuthorizationToken method (see Service Catalog):

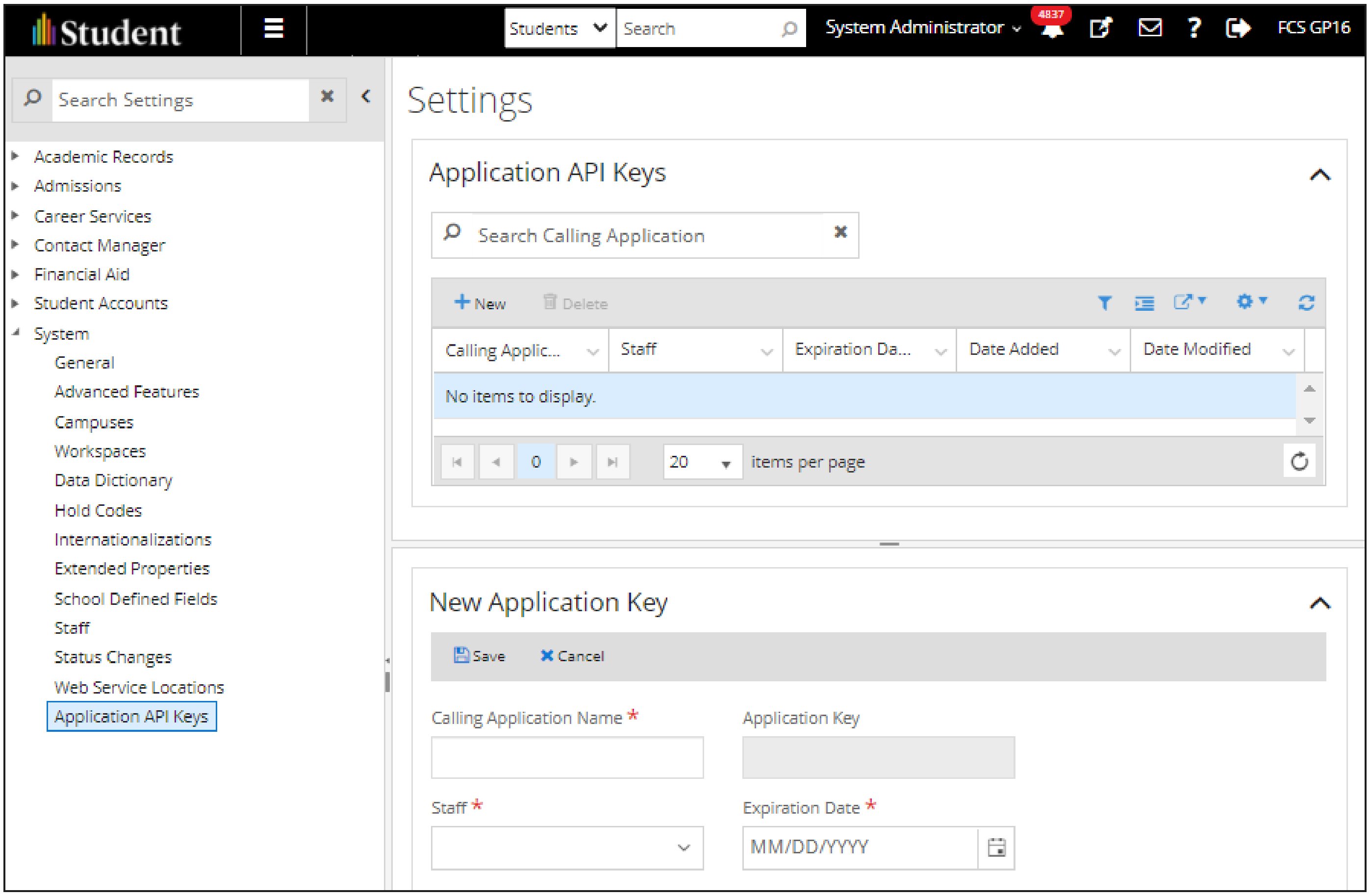
- IsKeyBasedSecurity (bool) – set to true if using key based security instead of supplying user-name/password
- AppKey (string) – plain text value of app key retrieved previously (e.g., you can assign this to the variable portalApplicationKey)
- CallingAppName (string) – name of the calling application. This needs to exactly match what's specified in the SyApplicationKey table (e.g., Anthology_Portal)

When IsKeyBasedSecurity is true and a valid AppKey is passed in, the Authentication Service validates the AppKey passed in against the CallingApplicationName and generates a token based on the AssociatedStaffId in the SyApplicationKey table. A token will be returned. Each system-generated application key will be associated with a system-generated staff user (e.g., PortalApiUser@anthology.local). This user only exists as a Staff record in the application and will not have a password or be required to be added as an AD/AAD user. As a result, this staff user will never be used to authenticate against. However, this user's authorizations in Anthology Student are used to determine permissions for API calls that are made in the system context, using the key. The user's Id is also used to update the database for the calls made by the client application. By default this user will have admin access. You can adjust the user's permission based on the needs of your application.

Anthology Student UI Updates

Anthology Student 22.0 provides a form for administrators to manage the CampusLink API key information. The form is located under Settings > System > Manage Application Keys.

Admin users can create a new record in the SyApplicationKey table, including generating the key. The encrypted value of the key will be stored in the ApplicationKey column. Once a new record is inserted, the plain text, decrypted key is shown in the form with a copy to clipboard option. Once the key is copied to clipboard (and shared with vendor securely), the key value is masked in the form.


Student
Students
Search
System Administrator
4837
FCS GP16
Search Settings
Academic Records
Admissions
Career Services
Contact Manager
Financial Aid
Student Accounts
System
General
Advanced Features
Campuses
Workspaces
Data Dictionary
Hold Codes
Internationalizations
Extended Properties
School Defined Fields
Staff
Status Changes
Web Service Locations
Application API Keys
Settings
Application API Keys
Search Calling Application
New
Delete
Calling Applic...
Staff
Expiration Da...
Date Added
Date Modified
No items to display.
0
20
items per page
New Application Key
Save
Cancel
Calling Application Name *
Application Key
Staff *
Expiration Date *
MM/DD/YYYY

Event Logs

The location of event logs depends on whether workflows are executed in a cloud environment or on premises.

Cloud Subscriptions

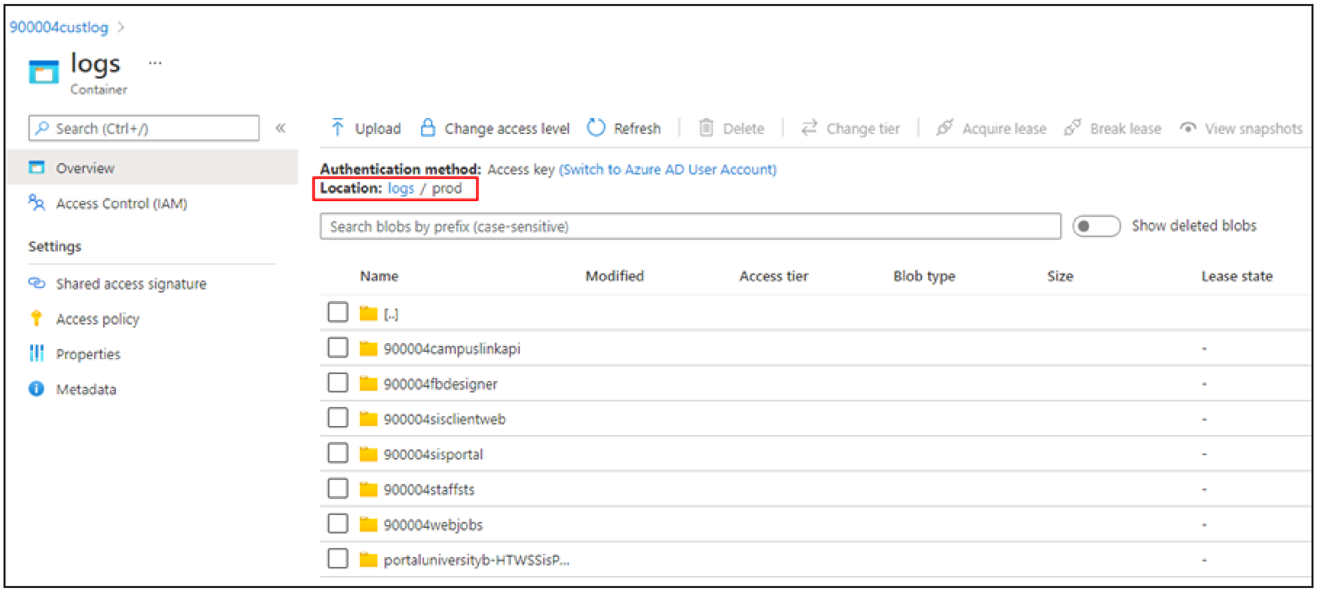
Azure blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data. Unstructured data is data that doesn't adhere to a particular data model or definition, such as text or binary data. See https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction.

The Azure blob storage container provides logs for all products included in your subscription, e.g., Anthology Student, Portal, Workflow Events, STS, CampusLink APIs, etc.

To find Azure blob storage logs for your cloud subscription:

1. Log in to the **Microsoft Azure Storage Explorer**.
2. Select your **subscription** (900004 in our example).
3. In the panel on the left side, navigate to **Storage Accounts** > **<your subscription>custlog** > **Blob Containers** > **logs**.
4. In the panel on the right side, open the **prod** directory.

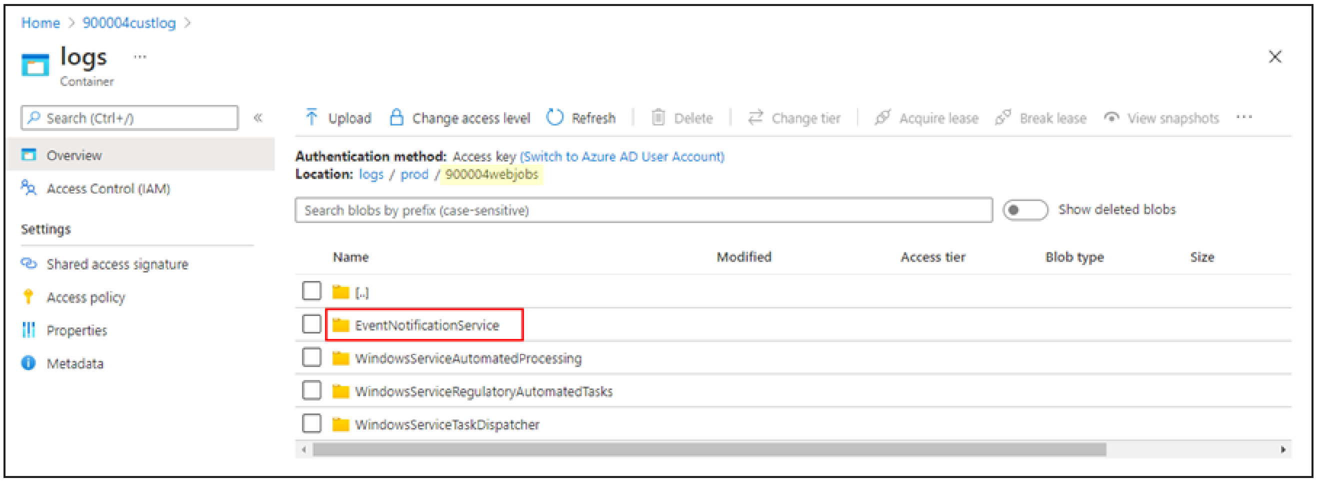
   Depending on your subscription, you will see folders for multiple products, e.g., CampusLinkAPI, Anthology Student (sisclientweb), Portal (sisportal), Form Designer, Forms Renderer, Staff STS, Web Jobs, etc. Select the folder for a product and navigate to the log you want to review. These folders contain the last 2 weeks of logs.



Authentication method: Access key (Switch to Azure AD User Account)
Location: logs / prod

Name	Modified	Access tier	Blob type	Size	Lease state
□ [..]					
□ 900004campuslinkapi					-
□ 900004fbdesigner					-
□ 900004sisclientweb					-
□ 900004sisportal					-
□ 900004staffsts					-
□ 900004webjobs					-
□ portaluniversityb-HTWSSisP...					-

5. For workflow logs, select the **<your subscription>webjobs** folder and then select **EventNotificationService**.



Authentication method: Access key (Switch to Azure AD User Account)
Location: logs / prod / 900004webjobs

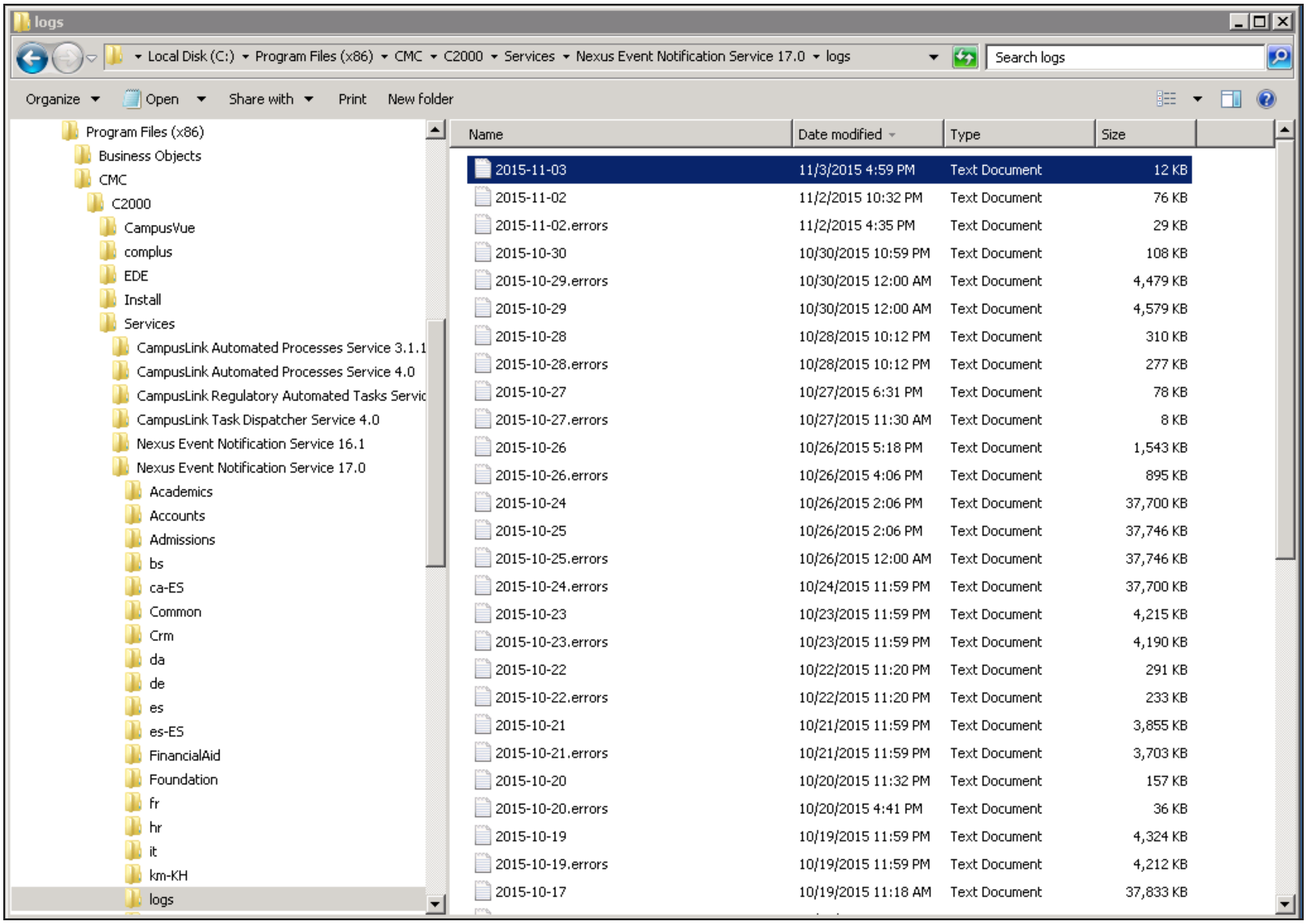
	Name	Modified	Access tier	Blob type	Size
□	[..]				
□	EventNotificationService				
□	WindowsServiceAutomatedProcessing				
□	WindowsServiceRegulatoryAutomatedTasks				
□	WindowsServiceTaskDispatcher				

View or download the workflow log you want to review.

On Premise Installations

Event logs for workflows that are executed on a Anthology Student server are written to the following folder on the server machine:

```
 Program Files (x86)\CMC\C2000\Services\Nexus Event Notification Service
<version>\logs.
```

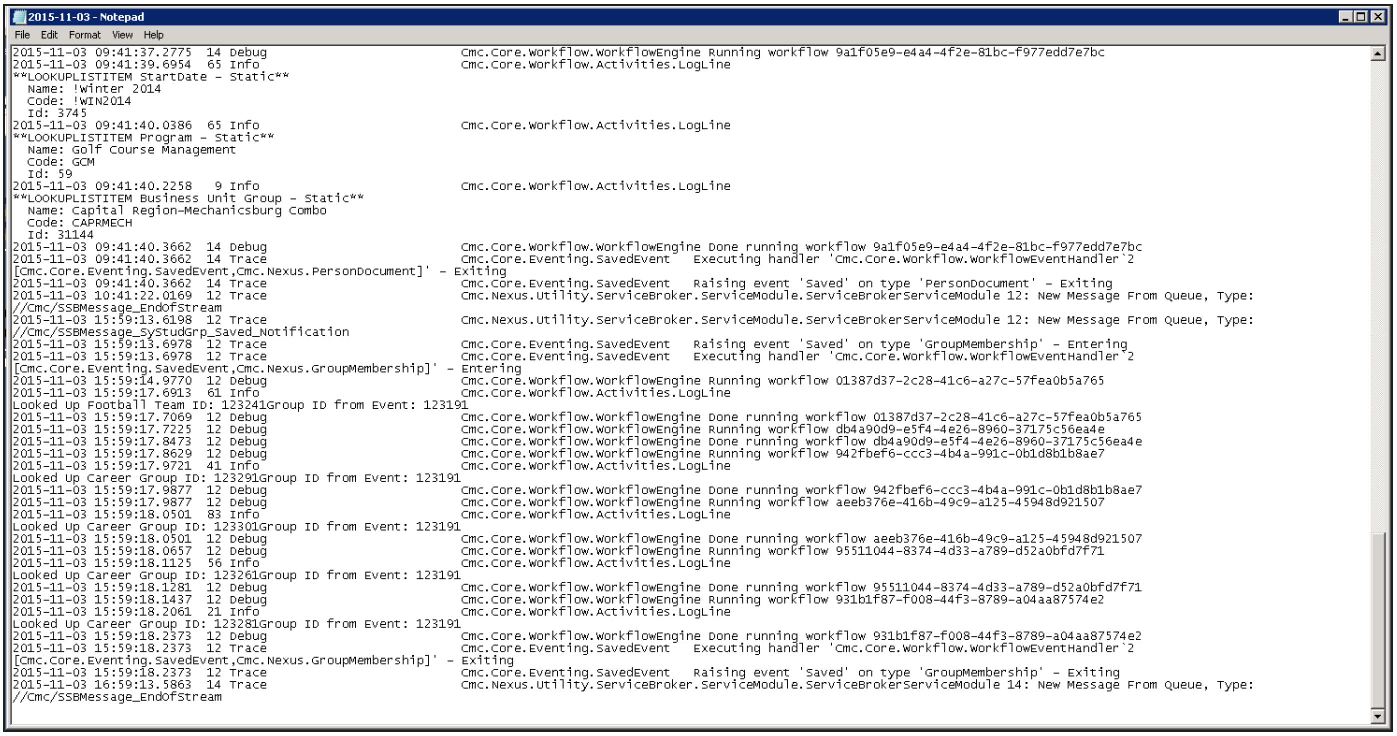


Local Disk (C:) ▸ Program Files (x86) ▸ CMC ▸ C2000 ▸ Services ▸ Nexus Event Notification Service 17.0 ▸ logs

- Program Files (x86)
  - Business Objects
  - CMC
    - C2000
      - CampusVue
      - complus
      - EDE
      - Install
      - Services
        - CampusLink Automated Processes Service 3.1.1
        - CampusLink Automated Processes Service 4.0
        - CampusLink Regulatory Automated Tasks Servic
        - CampusLink Task Dispatcher Service 4.0
        - Nexus Event Notification Service 16.1
        - Nexus Event Notification Service 17.0
          - Academics
          - Accounts
          - Admissions
          - bs
          - ca-ES
          - Common
          - Crm
          - da
          - de
          - es
          - es-ES
          - FinancialAid
          - Foundation
          - fr
          - hr
          - it
          - km-KH
          - logs

Name	Date modified	Type	Size
2015-11-03	11/3/2015 4:59 PM	Text Document	12 KB
2015-11-02	11/2/2015 10:32 PM	Text Document	76 KB
2015-11-02.errors	11/2/2015 4:35 PM	Text Document	29 KB
2015-10-30	10/30/2015 10:59 PM	Text Document	108 KB
2015-10-29.errors	10/30/2015 12:00 AM	Text Document	4,479 KB
2015-10-29	10/30/2015 12:00 AM	Text Document	4,579 KB
2015-10-28	10/28/2015 10:12 PM	Text Document	310 KB
2015-10-28.errors	10/28/2015 10:12 PM	Text Document	277 KB
2015-10-27	10/27/2015 6:31 PM	Text Document	78 KB
2015-10-27.errors	10/27/2015 11:30 AM	Text Document	8 KB
2015-10-26	10/26/2015 5:18 PM	Text Document	1,543 KB
2015-10-26.errors	10/26/2015 4:06 PM	Text Document	895 KB
2015-10-24	10/26/2015 2:06 PM	Text Document	37,700 KB
2015-10-25	10/26/2015 2:06 PM	Text Document	37,746 KB
2015-10-25.errors	10/26/2015 12:00 AM	Text Document	37,746 KB
2015-10-24.errors	10/24/2015 11:59 PM	Text Document	37,700 KB
2015-10-23	10/23/2015 11:59 PM	Text Document	4,215 KB
2015-10-23.errors	10/23/2015 11:59 PM	Text Document	4,190 KB
2015-10-22	10/22/2015 11:20 PM	Text Document	291 KB
2015-10-22.errors	10/22/2015 11:20 PM	Text Document	233 KB
2015-10-21	10/21/2015 11:59 PM	Text Document	3,855 KB
2015-10-21.errors	10/21/2015 11:59 PM	Text Document	3,703 KB
2015-10-20	10/20/2015 11:32 PM	Text Document	157 KB
2015-10-20.errors	10/20/2015 4:41 PM	Text Document	36 KB
2015-10-19	10/19/2015 11:59 PM	Text Document	4,324 KB
2015-10-19.errors	10/19/2015 11:59 PM	Text Document	4,212 KB
2015-10-17	10/19/2015 11:18 AM	Text Document	37,833 KB

The logs capture all workflow events including LogLine output, events associated with long running workflows, and errors captured by the Service Module Host.



```
2015-11-03 09:41:37.2775  14 Debug                          Cmc.Core.Workflow.WorkflowEngine Running workflow 9a1f05e9-e4a4-4f2e-81bc-f977edd7e7bc
2015-11-03 09:41:39.6954  65 Info                           Cmc.Core.Workflow.Activities.LogLine
**LOOKUPLISTITEM StartDate - Static**
  Name: !Winter 2014
  Code: !WIN2014
  Id: 3745
2015-11-03 09:41:40.0386  65 Info                           Cmc.Core.Workflow.Activities.LogLine
**LOOKUPLISTITEM Program - Static**
  Name: Golf Course Management
  Code: GCM
  Id: 59
2015-11-03 09:41:40.2258   9 Info                           Cmc.Core.Workflow.Activities.LogLine
**LOOKUPLISTITEM Business Unit Group - Static**
  Name: Capital Region-Mechanicsburg Combo
  Code: CAPRMECH
  Id: 31144
2015-11-03 09:41:40.3662  14 Debug                          Cmc.Core.Workflow.WorkflowEngine Done running workflow 9a1f05e9-e4a4-4f2e-81bc-f977edd7e7bc
2015-11-03 09:41:40.3662  14 Trace                          Cmc.Core.Eventing.SavedEvent   Executing handler 'Cmc.Core.Workflow.WorkflowEventHandler`2
[Cmc.Core.Eventing.SavedEvent,Cmc.Nexus.PersonDocument]' - Exiting
2015-11-03 09:41:40.3662  14 Trace                          Cmc.Core.Eventing.SavedEvent   Raising event 'Saved' on type 'PersonDocument' - Exiting
2015-11-03 10:41:22.0169  12 Trace                          Cmc.Nexus.Utility.ServiceBroker.ServiceModule.ServiceBrokerServiceModule 12: New Message From Queue, Type:
//Cmc/SSBMessage_EndofStream
2015-11-03 15:59:13.6198  12 Trace                          Cmc.Nexus.Utility.ServiceBroker.ServiceModule.ServiceBrokerServiceModule 12: New Message From Queue, Type:
//Cmc/SSBMessage_SyStudGrp_Saved_Notification
2015-11-03 15:59:13.6978  12 Trace                          Cmc.Core.Eventing.SavedEvent   Raising event 'Saved' on type 'GroupMembership' - Entering
2015-11-03 15:59:13.6978  12 Trace                          Cmc.Core.Eventing.SavedEvent   Executing handler 'Cmc.Core.Workflow.WorkflowEventHandler`2
[Cmc.Core.Eventing.SavedEvent,Cmc.Nexus.GroupMembership]' - Entering
2015-11-03 15:59:14.9770  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Running workflow 01387d37-2c28-41c6-a27c-57fea0b5a765
2015-11-03 15:59:17.6913  61 Info                           Cmc.Core.Workflow.Activities.LogLine
Looked Up Football Team ID: 123241Group ID from Event: 123191
2015-11-03 15:59:17.7069  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Done running workflow 01387d37-2c28-41c6-a27c-57fea0b5a765
2015-11-03 15:59:17.7225  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Running workflow db4a90d9-e5f4-4e26-8960-37175c56ea4e
2015-11-03 15:59:17.8473  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Done running workflow db4a90d9-e5f4-4e26-8960-37175c56ea4e
2015-11-03 15:59:17.8629  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Running workflow 942fbef6-ccc3-4b4a-991c-0b1d8b1b8ae7
2015-11-03 15:59:17.9721  41 Info                           Cmc.Core.Workflow.Activities.LogLine
Looked Up Career Group ID: 123291Group ID from Event: 123191
2015-11-03 15:59:17.9877  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Done running workflow 942fbef6-ccc3-4b4a-991c-0b1d8b1b8ae7
2015-11-03 15:59:17.9877  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Running workflow aeeb376e-416b-49c9-a125-45948d921507
2015-11-03 15:59:18.0501  83 Info                           Cmc.Core.Workflow.Activities.LogLine
Looked Up Career Group ID: 123301Group ID from Event: 123191
2015-11-03 15:59:18.0501  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Done running workflow aeeb376e-416b-49c9-a125-45948d921507
2015-11-03 15:59:18.0657  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Running workflow 95511044-8374-4d33-a789-d52a0bfd7f71
2015-11-03 15:59:18.1125  56 Info                           Cmc.Core.Workflow.Activities.LogLine
Looked Up Career Group ID: 123261Group ID from Event: 123191
2015-11-03 15:59:18.1281  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Done running workflow 95511044-8374-4d33-a789-d52a0bfd7f71
2015-11-03 15:59:18.1437  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Running workflow 931b1f87-f008-44f3-8789-a04aa87574e2
2015-11-03 15:59:18.2061  21 Info                           Cmc.Core.Workflow.Activities.LogLine
Looked Up Career Group ID: 123281Group ID from Event: 123191
2015-11-03 15:59:18.2373  12 Debug                          Cmc.Core.Workflow.WorkflowEngine Done running workflow 931b1f87-f008-44f3-8789-a04aa87574e2
2015-11-03 15:59:18.2373  12 Trace                          Cmc.Core.Eventing.SavedEvent   Executing handler 'Cmc.Core.Workflow.WorkflowEventHandler`2
[Cmc.Core.Eventing.SavedEvent,Cmc.Nexus.GroupMembership]' - Exiting
2015-11-03 15:59:18.2373  12 Trace                          Cmc.Core.Eventing.SavedEvent   Raising event 'Saved' on type 'GroupMembership' - Exiting
2015-11-03 16:59:13.5863  14 Trace                          Cmc.Nexus.Utility.ServiceBroker.ServiceModule.ServiceBrokerServiceModule 14: New Message From Queue, Type:
//Cmc/SSBMessage_EndofStream
```

The NLog settings determine the log level and target for event logs.

GitHub Repositories

Anthology Inc. has created a set of community-driven GitHub repositories to help share ideas, solutions, and knowledge about Anthology products.

For more information, download the attached PDF and refer to the following links:

Anthology GitHub Repositories	https://github.com/campusmanagement
Forms Builder Sequence Templates	https://github.com/campusmanagement/fb-sequence-templates
Workflow Samples	https://github.com/campusmanagement/workflow-samples
Integration Samples	https://github.com/campusmanagement/integration-samples
GitHub Resources	https://guides.github.com/

NLog

The default logging provider used by Anthology is NLog. NLog allows you to set up log targets, levels, rules, layouts, etc. through configuration.

Configure Logging

To configure logging, you need to modify the `nlog.config` file contained within the application's executing directory. For web applications, this file exists alongside the `web.config` file.

Note: In Workflow Composer 4.0.2 and later, the `nlog.config` file no longer exists. The NLog configuration settings have been placed in the app config file (WorkflowComposer.exe.config).

```
<?xml version="1.0" encoding="utf-8"?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <targets>
    <target name="file" xsi:type="File"
      layout="${longdate} ${threadid:padding=3} ${level:padding=-30} ${logger:padding=-30} ${message} ${exception:format=tostring}"
      fileName="${basedir}logs/${shortdate}.txt"
      keepFileOpen="true" />
    <target name="console" xsi:type="ColoredConsole"
      layout="${date:format=HH\:MM\:ss} ${threadid:padding=3} ${logger:padding=-30} ${message}" />
  </targets>
<rules>
  <logger name="*" minLevel="Error" writeTo="file" />
</rules>
</nlog>
```

Above is an example of a config file that is configured with two targets: file and console. The logging rules define which target is executed based on level (`Trace`, `Debug`, `Information`, `Warning`, `Error`, and `Fatal`). The configuration above logs to a subfolder off the base directory whenever an `Error` or `Fatal` level is logged by the application. To log verbose diagnostic information, you can change the `minLevel` to `Trace`, which will log all levels of diagnostic information.

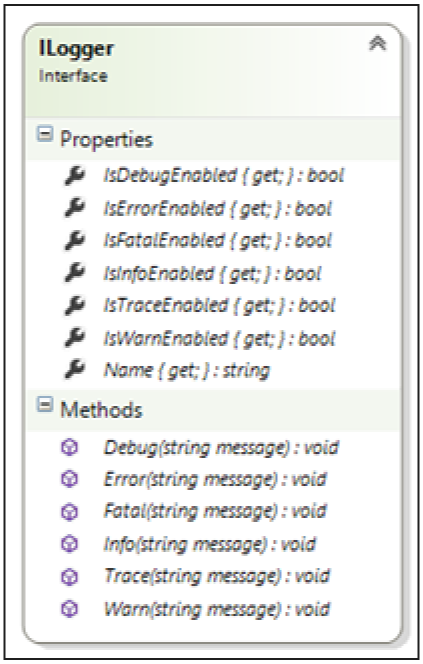
For additional information regarding the configuration file, see https://github.com/nlog/NLog/wiki/Configuration-file.

For supported NLog targets, see https://github.com/nlog/NLog/wiki/Targets.

Write Logs

Three public types are associated with the logging framework:

- ILoggerFactory
- ILogger
- LoggerExtensions (extensions methods for ILogger)



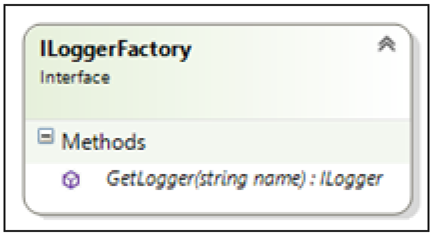
ILogger
Interface

Properties

IsDebugEnabled { get; } : bool
IsErrorEnabled { get; } : bool
IsFatalEnabled { get; } : bool
IsInfoEnabled { get; } : bool
IsTraceEnabled { get; } : bool
IsWarnEnabled { get; } : bool
Name { get; } : string

Methods

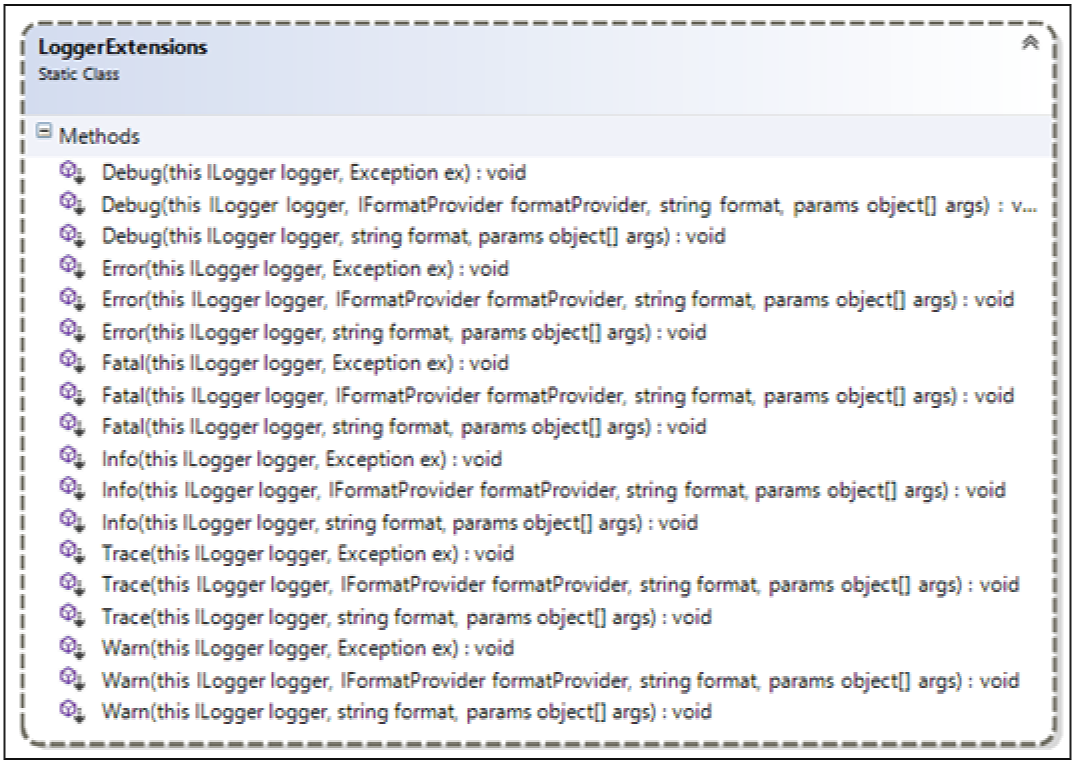
Debug(string message) : void
Error(string message) : void
Fatal(string message) : void
Info(string message) : void
Trace(string message) : void
Warn(string message) : void



ILoggerFactory
Interface

Methods

GetLogger(string name) : ILogger

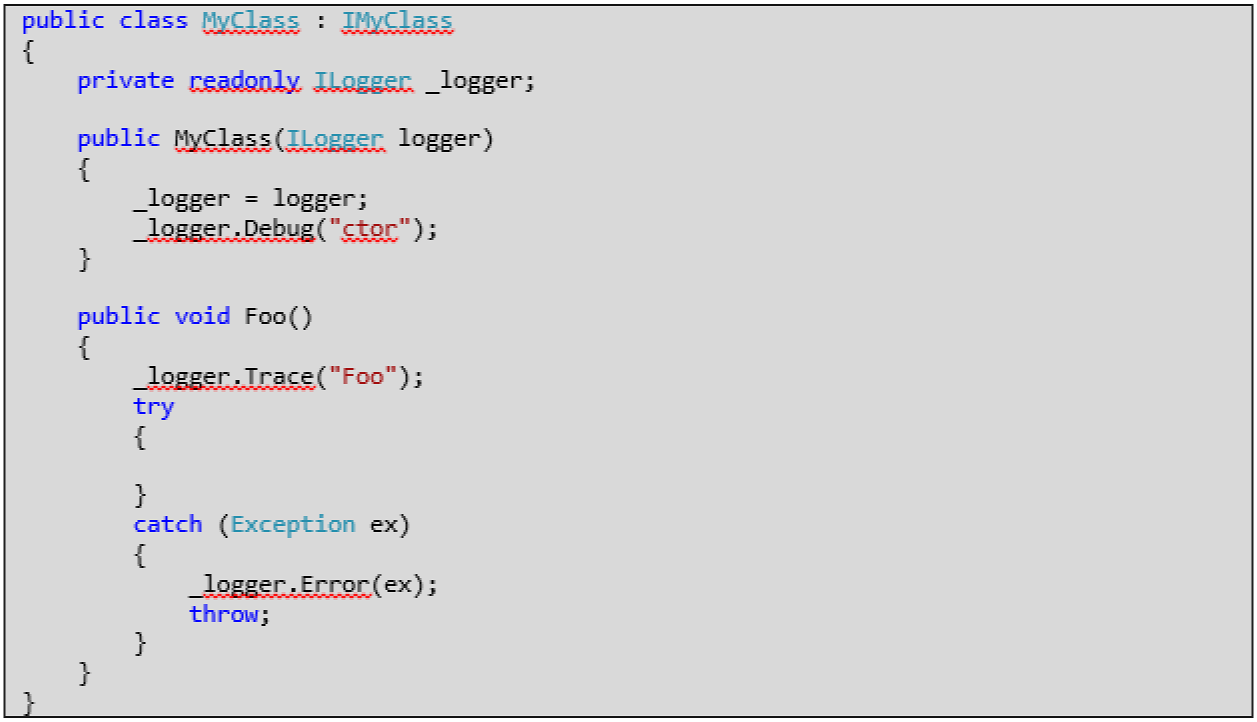


LoggerExtensions
Static Class

Methods

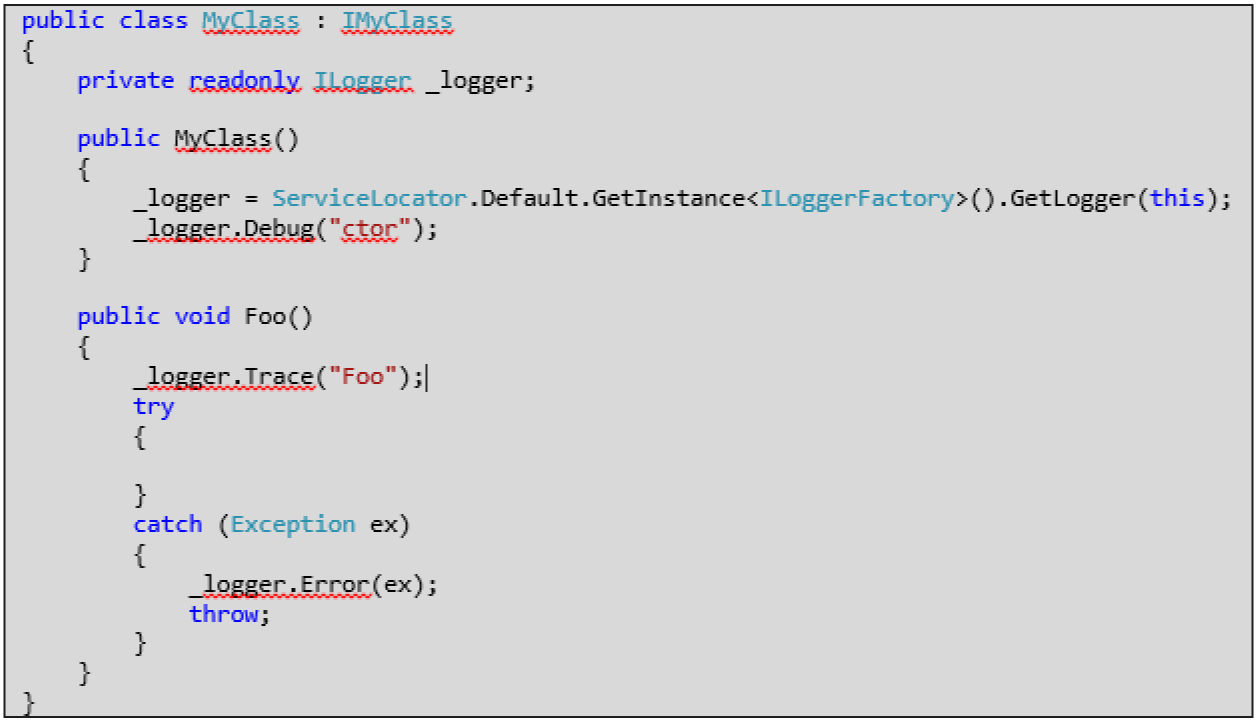
Debug(this ILogger logger, Exception ex) : void
Debug(this ILogger logger, IFormatProvider formatProvider, string format, params object[] args) : v...
Debug(this ILogger logger, string format, params object[] args) : void
Error(this ILogger logger, Exception ex) : void
Error(this ILogger logger, IFormatProvider formatProvider, string format, params object[] args) : void
Error(this ILogger logger, string format, params object[] args) : void
Fatal(this ILogger logger, Exception ex) : void
Fatal(this ILogger logger, IFormatProvider formatProvider, string format, params object[] args) : void
Fatal(this ILogger logger, string format, params object[] args) : void
Info(this ILogger logger, Exception ex) : void
Info(this ILogger logger, IFormatProvider formatProvider, string format, params object[] args) : void
Info(this ILogger logger, string format, params object[] args) : void
Trace(this ILogger logger, Exception ex) : void
Trace(this ILogger logger, IFormatProvider formatProvider, string format, params object[] args) : void
Trace(this ILogger logger, string format, params object[] args) : void
Warn(this ILogger logger, Exception ex) : void
Warn(this ILogger logger, IFormatProvider formatProvider, string format, params object[] args) : void
Warn(this ILogger logger, string format, params object[] args) : void

There are two ways to enable logging in your class. The preferred way is to receive an ILogger interface as a constructor dependency. The IoC container ensures that this dependency is wired for you.



```
public class MyClass : IMyClass
{
    private readonly ILogger _logger;

    public MyClass(ILogger logger)
    {
        _logger = logger;
        _logger.Debug("ctor");
    }

    public void Foo()
    {
        _logger.Trace("Foo");
        try
        {

        }
        catch (Exception ex)
        {
            _logger.Error(ex);
            throw;
        }
    }
}
```

If your class is a legacy class that does not support DI, you can use the ServiceLocator to retrieve an ILoggerFactory to create the logger.



```
public class MyClass : IMyClass
{
    private readonly ILogger _logger;

    public MyClass()
    {
        _logger = ServiceLocator.Default.GetInstance<ILoggerFactory>().GetLogger(this);
        _logger.Debug("ctor");
    }

    public void Foo()
    {
        _logger.Trace("Foo");
        try
        {

        }
        catch (Exception ex)
        {
            _logger.Error(ex);
            throw;
        }
    }
}
```

Add Log Messages to Classes

Once you have a logger in a class, it is important to add the relevant LOG messages to it that will help us all in debugging and understanding how this class is behaving.

Log Non-Exception Messages

Trace Messages

Use these messages to trace which lines of source code are being executed; they will log what is going on with the code.

Usage: `_log.Trace(“Your message.”)`

Debug Messages

Use these messages to output the contents or values of variables or properties during the execution of source code; they will log the important values of objects that may affect how the code will execute.

Usage: `_log.Debug(“Your message. variable1={0}.”, variable1)`

Info Messages

Use these messages to log information that may be useful to know about the normal operation of the application (such as environment variables, paths, etc.).

Usage: `_log.Info(“Your message. variable1={0}.”, variable1)`

Warning Messages

Use these messages to log messages that we are not sure are acceptable or to track variable/property values that may be close to being out of the acceptable range.

Usage: `_log.Warn(“Your message. variable1={0}.”, variable1)`

Error Messages

Use these messages to log any exceptions we have that are not being handled. This is typically used in the CATCH of a TRY/CATCH block.

Usage: See Log Exception Messages.

Fatal Messages

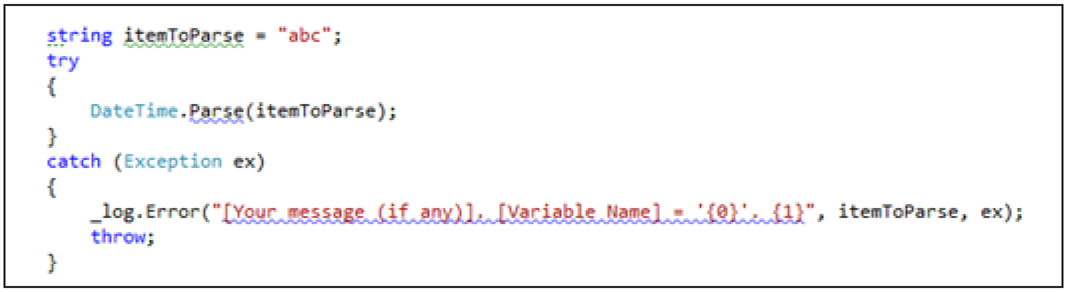
Use these messages to log special conditions that indicate that something went terribly wrong in the execution of the code.

Usage: See Log Exception Messages.

Log Exception Messages

To properly log an exception, you should follow one of the patterns shown below. This will allow you to capture the full exception details and also include (if necessary) any other values that may be important for debugging.

Scenario 1: Log a custom message, a variable value, and an exception

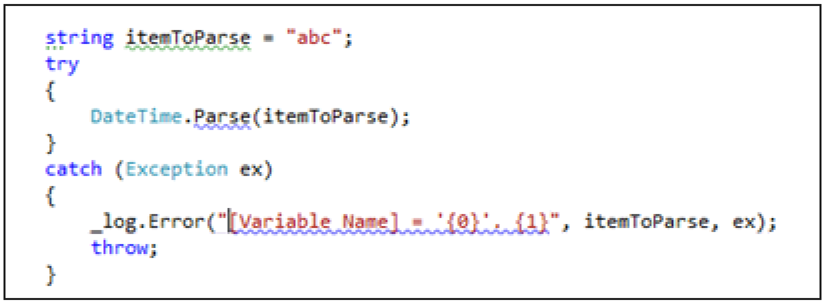


```
string itemToParse = "abc";
try
{
    DateTime.Parse(itemToParse);
}
catch (Exception ex)
{
    _log.Error("[Your message (if any)]. [Variable Name] = '{0}'. {1}", itemToParse, ex);
    throw;
}
```

Result log message:

[Your message (if any)]. [Variable Name] = 'abc'. System.FormatException: The string was not recognized as a valid DateTime. There is an unknown word starting at index 0. at System.DateTime.Parse(String s) at Cmc.UI.Web.EcoSysW3C.------() in \DEV\DEV\Cmc\UI\Web\Cmc.UI.Web.EcoSysW3C\------.cs:line xx

Scenario 2: Log a variable value and an exception

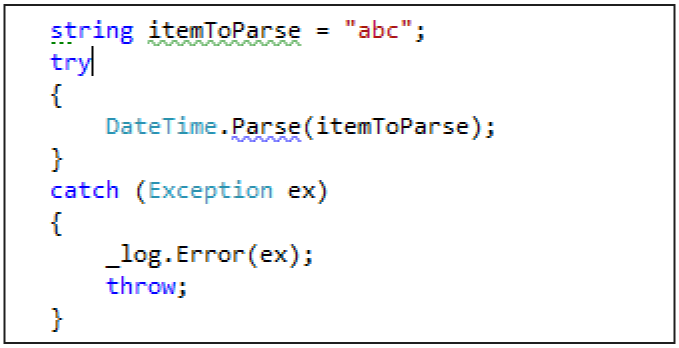


```
string itemToParse = "abc";
try
{
    DateTime.Parse(itemToParse);
}
catch (Exception ex)
{
    _log.Error("[Variable Name] = '{0}'. {1}", itemToParse, ex);
    throw;
}
```

Result log message:

[Variable Name] = 'abc'. System.FormatException: The string was not recognized as a valid DateTime. There is an unknown word starting at index 0. at System.DateTime.Parse(String s) at Cmc.UI.Web.EcoSysW3C.------() in \DEV\DEV\Cmc\UI\Web\Cmc.UI.Web.EcoSysW3C\------.cs:line xx

Scenario 3: Log only an exception



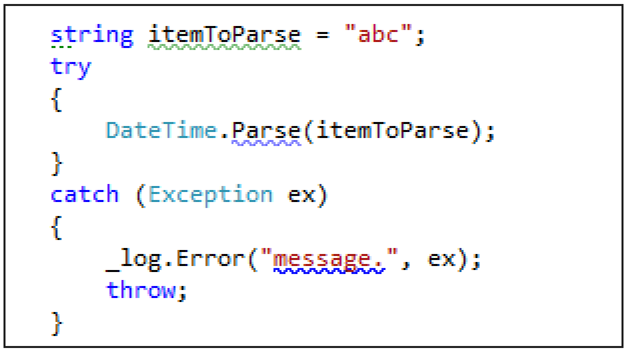
```
string itemToParse = "abc";
try
{
    DateTime.Parse(itemToParse);
}
catch (Exception ex)
{
    _log.Error(ex);
    throw;
}
```

Result log message:

System.FormatException: The string was not recognized as a valid DateTime. There is an unknown word starting at index 0. at System.DateTime.Parse(String s) at Cmc.UI.Web.EcoSysW3C.------() in \DEV\DEV\Cm-c\UI\Web\Cmc.UI.Web.EcoSysW3C\------.cs:line xx

Note: You must always inject the exception to the string message using {0}!

If you log an exception as shown below, it will fail to include the exception in the log message. See result of this message below:



```
string itemToParse = "abc";
try
{
    DateTime.Parse(itemToParse);
}
catch (Exception ex)
{
    _log.Error("message.", ex);
    throw;
}
```

Result log message:

message

Read Log Messages to Debug or Troubleshoot

There are three different ways to see your log messages when you wish to debug or troubleshoot an issue:

1. Access the SQL server and get values from the LOGS table (if they are being logged to the DB)
2. Access the local log files being saved in (webroot)/LOGS
3. Use a real-time viewer

You can download the FREE LOG viewer from: http://www.legitlog.com/Products/LegitLogViewer.

Legit Log™

Once you install it, you can use it to:

- Read the log text file, or
- View messages in real-time as they are added to the logger.

To enable real-time logging, follow these steps:

1. Select **Logs >> Live Capture Log**.
2. Select **Start capture global**.

You should now start seeing any log messages as they are added into the logger.

For additional information, see the NLog website: http://nlog-project.org.

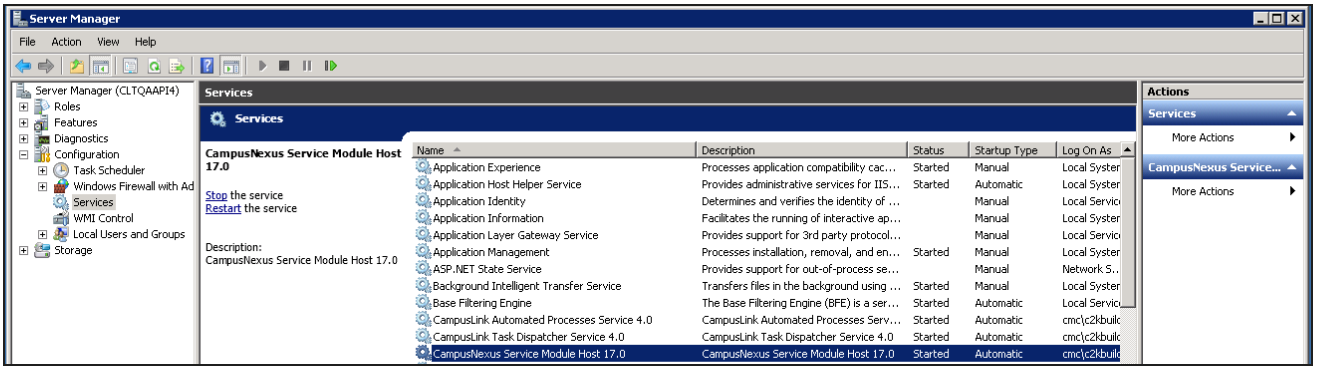
Service Module Host

ServiceModuleHost.exe is a Windows service that allows Saved Events to execute and is responsible for hosting plugin modules to simplify deployment and maintenance of processes that run in the background. Installation Manager sets the services to be started automatically; however, when you are building workflows, it is important to ensure that the Anthology Service Module Host is running on the server.

Stop/Start the Service Module Host Service

1. On the server where the workflows are executed, select **Start > Administrative Tools > Server Manager**, right-click and select **Run as administrator**.
2. Navigate to **Configuration > Services** and select the **Anthology Service Module Host** service.

   By default, the Startup Type of the Anthology Service Module Host is set to **Automatic** with a Status of **Started**.

   CampusNexus Service Module Host 17.0

   Stop the service
   Restart the service

   Description:
   CampusNexus Service Module Host 17.0

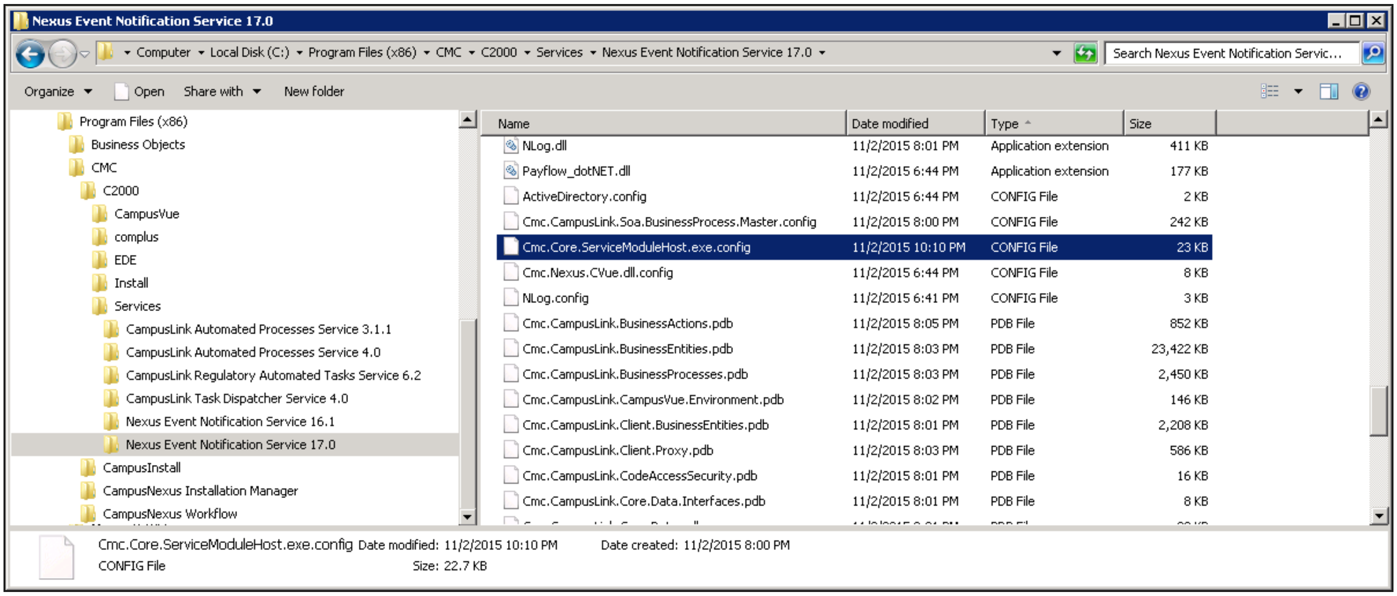
Name	Description	Status	Startup Type	Log On As
Application Experience	Processes application compatibility cac...	Started	Manual	Local Syster
Application Host Helper Service	Provides administrative services for IIS...	Started	Automatic	Local Syster
Application Identity	Determines and verifies the identity of ...		Manual	Local Servic
Application Information	Facilitates the running of interactive ap...		Manual	Local Syster
Application Layer Gateway Service	Provides support for 3rd party protocol...		Manual	Local Servic
Application Management	Processes installation, removal, and en...	Started	Manual	Local Syster
ASP.NET State Service	Provides support for out-of-process se...		Manual	Network S..
Background Intelligent Transfer Service	Transfers files in the background using ...	Started	Manual	Local Syster
Base Filtering Engine	The Base Filtering Engine (BFE) is a ser...	Started	Automatic	Local Servic
CampusLink Automated Processes Service 4.0	CampusLink Automated Processes Serv...	Started	Automatic	cmc\c2kbuilc
CampusLink Task Dispatcher Service 4.0	CampusLink Task Dispatcher Service 4.0	Started	Automatic	cmc\c2kbuilc
CampusNexus Service Module Host 17.0	CampusNexus Service Module Host 17.0	Started	Automatic	cmc\c2kbuilc

3. To stop or restart the service, click **Stop** or **Restart** the service.

Service Module Host Config File

Installation Manager updates the configuration files to ensure that they point to the correct database and contain proper settings. The configuration file for the ServiceModuleHost.exe and normally does not need to be modified; however, you should be aware of the SQL Reconnect Setting and Connection Strings.

The Service Module Host config file is located in C:\Program Files (x86)\CMC\C2000\Services\Nexus Event Notification Service <version>.



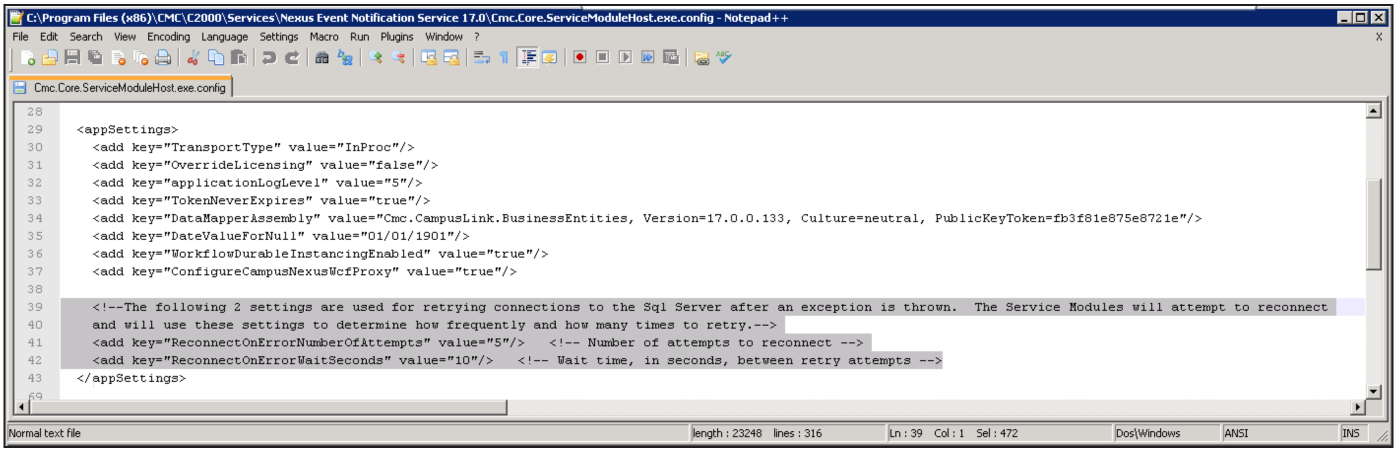
Nexus Event Notification Service 17.0
Computer ▸ Local Disk (C:) ▸ Program Files (x86) ▸ CMC ▸ C2000 ▸ Services ▸ Nexus Event Notification Service 17.0
Search Nexus Event Notification Servic...
Organize
Open
Share with
New folder
Program Files (x86)
Business Objects
CMC
C2000
CampusVue
complus
EDE
Install
Services
CampusLink Automated Processes Service 3.1.1
CampusLink Automated Processes Service 4.0
CampusLink Regulatory Automated Tasks Service 6.2
CampusLink Task Dispatcher Service 4.0
Nexus Event Notification Service 16.1
Nexus Event Notification Service 17.0
CampusInstall
CampusNexus Installation Manager
CampusNexus Workflow

Name	Date modified	Type	Size
NLog.dll	11/2/2015 8:01 PM	Application extension	411 KB
Payflow_dotNET.dll	11/2/2015 6:44 PM	Application extension	177 KB
ActiveDirectory.config	11/2/2015 6:44 PM	CONFIG File	2 KB
Cmc.CampusLink.Soa.BusinessProcess.Master.config	11/2/2015 8:00 PM	CONFIG File	242 KB
Cmc.Core.ServiceModuleHost.exe.config	11/2/2015 10:10 PM	CONFIG File	23 KB
Cmc.Nexus.CVue.dll.config	11/2/2015 6:44 PM	CONFIG File	8 KB
NLog.config	11/2/2015 6:41 PM	CONFIG File	3 KB
Cmc.CampusLink.BusinessActions.pdb	11/2/2015 8:05 PM	PDB File	852 KB
Cmc.CampusLink.BusinessEntities.pdb	11/2/2015 8:03 PM	PDB File	23,422 KB
Cmc.CampusLink.BusinessProcesses.pdb	11/2/2015 8:03 PM	PDB File	2,450 KB
Cmc.CampusLink.CampusVue.Environment.pdb	11/2/2015 8:02 PM	PDB File	146 KB
Cmc.CampusLink.Client.BusinessEntities.pdb	11/2/2015 8:01 PM	PDB File	2,208 KB
Cmc.CampusLink.Client.Proxy.pdb	11/2/2015 8:03 PM	PDB File	586 KB
Cmc.CampusLink.CodeAccessSecurity.pdb	11/2/2015 8:01 PM	PDB File	16 KB
Cmc.CampusLink.Core.Data.Interfaces.pdb	11/2/2015 8:01 PM	PDB File	8 KB

Cmc.Core.ServiceModuleHost.exe.config Date modified: 11/2/2015 10:10 PM Date created: 11/2/2015 8:00 PM
CONFIG File Size: 22.7 KB

SQL Reconnect Setting

The Service Module Host service has logic to limit the reconnection attempts when the Service Module Host service senses a connection failure to the SQL database. The time duration is a configured value in seconds that the Service Module Host service uses to attempt the connection again. The settings contain a Number of Retries value indicating how many times to retry the connection.



C:\Program Files (x86)\CMC\C2000\Services\Nexus Event Notification Service 17.0\Cmc.Core.ServiceModuleHost.exe.config - Notepad++
File
Edit
Search
View
Encoding
Language
Settings
Macro
Run
Plugins
Window
?
Cmc.Core.ServiceModuleHost.exe.config

```

  <appSettings>
    <add key="TransportType" value="InProc"/>
    <add key="OverrideLicensing" value="false"/>
    <add key="applicationLogLevel" value="5"/>
    <add key="TokenNeverExpires" value="true"/>
    <add key="DataMapperAssembly" value="Cmc.CampusLink.BusinessEntities, Version=17.0.0.133, Culture=neutral, PublicKeyToken=fb3f81e875e8721e"/>
    <add key="DateValueForNull" value="01/01/1901"/>
    <add key="WorkflowDurableInstancingEnabled" value="true"/>
    <add key="ConfigureCampusNexusWcfProxy" value="true"/>

    <!--The following 2 settings are used for retrying connections to the Sql Server after an exception is thrown.  The Service Modules will attempt to reconnect
    and will use these settings to determine how frequently and how many times to retry.-->
    <add key="ReconnectOnErrorNumberOfAttempts" value="5"/>    <!-- Number of attempts to reconnect -->
    <add key="ReconnectOnErrorWaitSeconds" value="10"/>    <!-- Wait time, in seconds, between retry attempts -->
  </appSettings>
```

Normal text file   length : 23248   lines : 316   Ln : 39   Col : 1   Sel : 472   Dos\Windows   ANSI   INS

If, after the number of attempts have been tried and the SQL server is still unavailable, the Service Module Host logs a fatal exception indicating that the Windows service should be restarted after the SQL connection issue has been resolved. The Service Module Host then needs to be stopped and restarted to re-establish the connection (see Stop/Start the Service Module Host Service).

The following is an example of an error displayed in the workflow Event Log when the timeout expired and a reconnection was attempted:

2015-08-29 00:00:04.7756 13 Error

Cmc.Nexus.Utility.ServiceBroker.ServiceModule.ServiceBrokerServiceModule System.InvalidOperationException: Timeout expired. The timeout period elapsed prior to obtaining a connection from the pool. This may have occurred because all pooled connections were in use and max pool size was reached.

at System.Data.ProviderBase.DbConnectionFactory.TryGetConnection(DbConnection owningConnection, TaskCompletionSource`1 retry, DbConnectionOptions userOptions, DbConnectionInternal oldConnection, DbConnectionInternal& connection)

at System.Data.ProviderBase.DbConnectionInternal.TryOpenConnectionInternal(DbConnection outerConnection, DbConnectionFactory connectionFactory, TaskCompletionSource`1 retry, DbConnectionOptions userOptions)

at System.Data.ProviderBase.DbConnectionClosed.TryOpenConnection(DbConnection outerConnection, DbConnectionFactory connectionFactory, TaskCompletionSource`1 retry, DbConnectionOptions userOptions)

at System.Data.SqlClient.SqlConnection.TryOpenInner(TaskCompletionSource`1 retry)

at System.Data.SqlClient.SqlConnection.TryOpen(TaskCompletionSource`1 retry)

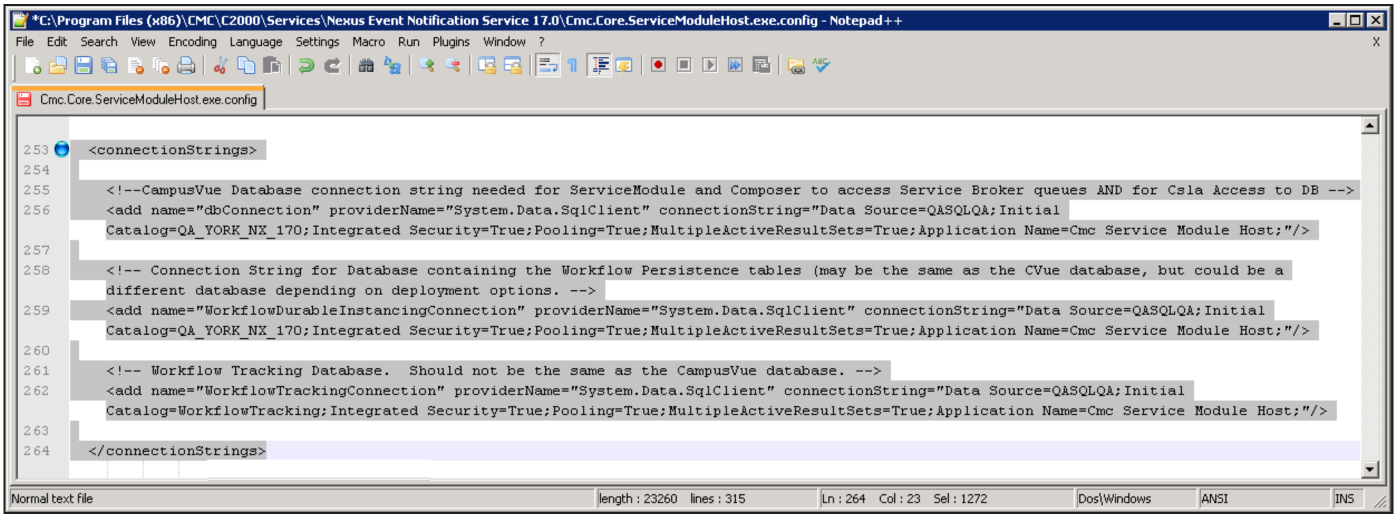
at System.Data.SqlClient.SqlConnection.Open()

If errors like this occur frequently and fill up the event logs, you might need to adjust the values for **ReconnectOnErrorNumberOfAttempts** (default value = 5) and **ReconnectOnErrorWaitSeconds** (default value = 10) in the config file of the Service Module Host.

Connection Strings

The config file of the Service Module Host contains connection strings for the following databases:

- Anthology Student Database
- Database containing the workflow persistence tables
- Workflow Tracking Database


*C:\Program Files (x86)\CMC\C2000\Services\Nexus Event Notification Service 17.0\Cmc.Core.ServiceModuleHost.exe.config - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Cmc.Core.ServiceModuleHost.exe.config
Normal text file
length : 23260 lines : 315
Ln : 264 Col : 23 Sel : 1272
Dos\Windows
ANSI
INS

```
  <connectionStrings>

    <!--CampusVue Database connection string needed for ServiceModule and Composer to access Service Broker queues AND for Csla Access to DB -->
    <add name="dbConnection" providerName="System.Data.SqlClient" connectionString="Data Source=QASQLQA;Initial
    Catalog=QA_YORK_NX_170;Integrated Security=True;Pooling=True;MultipleActiveResultSets=True;Application Name=Cmc Service Module Host;"/>

    <!-- Connection String for Database containing the Workflow Persistence tables (may be the same as the CVue database, but could be a
    different database depending on deployment options. -->
    <add name="WorkflowDurableInstancingConnection" providerName="System.Data.SqlClient" connectionString="Data Source=QASQLQA;Initial
    Catalog=QA_YORK_NX_170;Integrated Security=True;Pooling=True;MultipleActiveResultSets=True;Application Name=Cmc Service Module Host;"/>

    <!-- Workflow Tracking Database.  Should not be the same as the CampusVue database. -->
    <add name="WorkflowTrackingConnection" providerName="System.Data.SqlClient" connectionString="Data Source=QASQLQA;Initial
    Catalog=WorkflowTracking;Integrated Security=True;Pooling=True;MultipleActiveResultSets=True;Application Name=Cmc Service Module Host;"/>

  </connectionStrings>
```

The connection strings enable workflow tracking and persisted workflows.

The Anthology Student Database connection string is specifically referenced in the following workflow activities:

- ExecuteDataReader
- ExecuteNonQuery
- ExecuteQuery

⚠ In general, the connection strings used during workflow execution are retrieved from the web.config of the product that triggers workflow execution.

Only if you want to run a workflow with ExecuteDataReader, ExecuteNonQuery, or ExecuteQuery activity in test mode using the **Run** option in Workflow Composer, would you need to manually add the connection string to the Workflow Composer web.config file.

Workflow Tracking DB Cleanup Script

If you are using the Workflow Tracking database, you may find that it grows at a rapid pace depending on the configured tracking level.

The attached script can be run against the tracking database to clean out records on a regular basis. The steps below describe the parameter that needs to be entered and what is needed to schedule it as an SQL job.

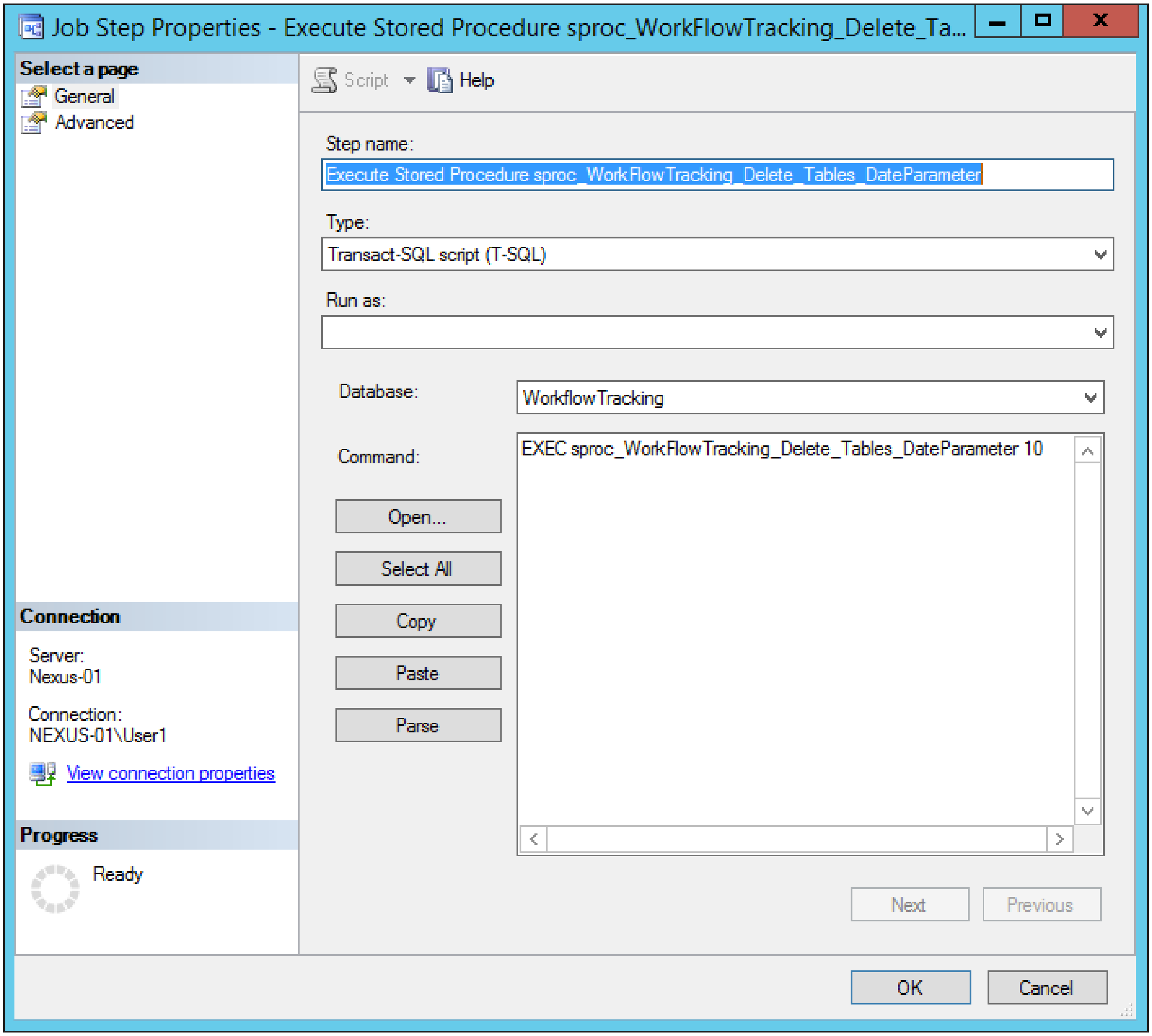
1. Use the script sproc_WorkFlowTracking_Delete_Tables_DateParameter.sql. Download or copy it below.

   The script only requires a **date** parameter to be populated. In the scenario below, anything older than **10** days would be deleted.

```
IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.ROUTINES r WHERE r.routine_name='sproc_
WorkFlowTracking_Delete_Tables_DateParameter' and r.routine_schema='dbo')
    DROP PROCEDURE dbo.sproc_WorkFlowTracking_Delete_Tables_DateParameter
GO

/****** Object:  StoredProcedure [dbo].[sproc_WorkFlowTracking_Delete_Tables_DateParameter]
Script Date: 10/9/2015 10:42:47 AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE    [dbo].[sproc_WorkFlowTracking_Delete_Tables_DateParameter]
 -- Add the parameters for the stored procedure here
 @NumberOfDays int
AS
BEGIN

/*
Exec [dbo].[sproc_WorkFlowTracking_Delete_Tables_DateParameter]  10
*/
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    DECLARE @HowManyRecordsTobeDeleted INT;
    DECLARE @InitialSet INT;
    Set @InitialSet = 500;
    SET @HowManyRecordsTobeDeleted = @InitialSet;

    WHILE @HowManyRecordsTobeDeleted > 10
    BEGIN

        BEGIN TRY
            BEGIN TRAN

              delete  top (@HowManyRecordsTobeDeleted) [workflowtracking].[System.Workflow.Tracking].[WorkflowInstanceEventsTable]  from [workflowtracking].[System.Workflow.Tracking].[WorkflowInstanceEventsTable]
              where  (DATEDIFF(day,[workflowtracking].[System.Workflow.Tracking].[WorkflowInstanceEventsTable].TimeCreated,Getdate())>=@NumberOfDays)
            SET @HowManyRecordsTobeDeleted = @@ROWCOUNT
            print 'deleted WorkflowInstanceEventsTable'
            COMMIT TRAN
         END TRY
         BEGIN CATCH
            ROLLBACK TRAN
            set @HowManyRecordsTobeDeleted = 0
            print 'ERROR in deleting WorkflowInstanceEventsTable'
        END CATCH

    END
    set @HowManyRecordsTobeDeleted = @InitialSet
    WHILE @HowManyRecordsTobeDeleted > 0
    BEGIN

        BEGIN TRY
            BEGIN TRAN

              delete top (@HowManyRecordsTobeDeleted) [workflowtracking].[System.Workflow.Tracking].[ActivityInstanceEventsTable]  from [workflowtracking].[System.Workflow.Tracking].[ActivityInstanceEventsTable] st
              where    (DATEDIFF(day,st.TimeCreated,Getdate())>=@NumberOfDays)
            SET @HowManyRecordsTobeDeleted = @@ROWCOUNT
            print 'deleted ActivityInstanceEventsTable'
            COMMIT TRAN
          END TRY
         BEGIN CATCH
            ROLLBACK TRAN
            set @HowManyRecordsTobeDeleted = 0
            print 'ERROR in deleting ActivityInstanceEventsTable'
        END CATCH

    END

    set @HowManyRecordsTobeDeleted = @InitialSet
    WHILE @HowManyRecordsTobeDeleted > 0
    BEGIN
        BEGIN TRY
            BEGIN TRAN

                delete  top (@HowManyRecordsTobeDeleted) [workflowtracking].[System.Workflow.Tracking].
[ExtendedActivityEventsTable] from [workflowtracking].[System.Workflow.Tracking].[ExtendedActivityEventsTable]   stc
              where (DATEDIFF(day,stc.TimeCreated,Getdate())>=@NumberOfDays)
            SET @HowManyRecordsTobeDeleted = @@ROWCOUNT
            print 'deleted ExtendedActivityEventsTable'
            COMMIT TRAN
            END TRY
         BEGIN CATCH
            ROLLBACK TRAN
            set @HowManyRecordsTobeDeleted = 0
            print 'ERROR in deleting ExtendedActivityEventsTable'
        END CATCH

    END

    set @HowManyRecordsTobeDeleted = @InitialSet
    WHILE @HowManyRecordsTobeDeleted > 0
    BEGIN
        BEGIN TRY
            BEGIN TRAN

                delete  top (@HowManyRecordsTobeDeleted) [workflowtracking].[System.Workflow.Tracking].[Bo
markResumptionEventsTable] from [workflowtracking].[System.Workflow.Tracking].[BookmarkResumptionEventsTable]   stc
              where (DATEDIFF(day,stc.TimeCreated,Getdate())>=@NumberOfDays)
            SET @HowManyRecordsTobeDeleted = @@ROWCOUNT
            print 'deleted BookmarkResumptionEventsTable'
            COMMIT TRAN
            END TRY
         BEGIN CATCH
```

```
            ROLLBACK TRAN
            set @HowManyRecordsTobeDeleted = 0
            print 'ERROR in deleting BookmarkResumptionEventsTable'
        END CATCH

    END

    set @HowManyRecordsTobeDeleted = @InitialSet
    WHILE @HowManyRecordsTobeDeleted > 0
    BEGIN
        BEGIN TRY
            BEGIN TRAN

                delete  top (@HowManyRecordsTobeDeleted) [workflowtracking].[System.Workflow.Tracking].[Cu
tomTrackingEventsTable] from [workflowtracking].[System.Workflow.Tracking].[Cus-
tomTrackingEventsTable]   stc
              where (DATEDIFF(day,stc.TimeCreated,Getdate())>=@NumberOfDays)
            SET @HowManyRecordsTobeDeleted = @@ROWCOUNT
            print 'deleted CustomTrackingEventsTable'
            COMMIT TRAN
            END TRY
          BEGIN CATCH
            ROLLBACK TRAN
            set @HowManyRecordsTobeDeleted = 0
            print 'ERROR in deleting CustomTrackingEventsTable'
        END CATCH

    END

END

GO
```

2. The script can also be scheduled as an SQL job to run based on a schedule.



Job Step Properties - Execute Stored Procedure sproc_WorkFlowTracking_Delete_Ta...

Step name: Execute Stored Procedure sproc_WorkFlowTracking_Delete_Tables_DateParameter

Type: Transact-SQL script (T-SQL)

Run as:

Database: WorkflowTracking

Command: EXEC sproc_WorkFlowTracking_Delete_Tables_DateParameter 10

Server: Nexus-01

Connection: NEXUS-01\User1

Progress: Ready

3. You can control the amount of data being tracked by using trackingProfiles (defined within the Service Module Host config file).

   Notes:

   - If tracking is configured to track variables, this database can grow extremely fast.
   - If you do not want tracking enabled, you can remove the tracking profile from the config file.
   - If you simply want to track the start and stop of a workflow, we recommend the following setting:

     ```
     <system.serviceModel>
         <tracking>
           <profiles>
             <trackingProfile name="DefaultTrackingProfile">
     ```

```
        <workflow activityDefinitionId="*">
          <workflowInstanceQueries>
            <workflowInstanceQuery>
              <states>
                <state name="Started" />
                <state name="Completed" />
              </states>
            </workflowInstanceQuery>
          </workflowInstanceQueries>
        </workflow>
      </trackingProfile>
    </profiles>
  </tracking>
</system.serviceModel>
```