Table of Contents

Introduction ........................................................................................................................................... 3

Handling the Full Range of Project, Portfolio, and Collaboration Workflows .................................. 3

Demand Management Workflows .................................................................................................... 3

Event Driven Workflows .................................................................................................................. 3

Collaboration Workflows .................................................................................................................. 4

Overview of Event Handlers in Project Server 2010 ............................................................................ 5

Why Attach Workflows to Events? ...................................................................................................... 5

Creating an Event Driven Workflow to Automatically Backup Projects to the Archive Database.... 6

Building the archive workflow ........................................................................................................... 6

Associating the workflow with an event ............................................................................................. 19

Running the Workflow ........................................................................................................................ 21

Viewing the status of a workflow ........................................................................................................ 22

Conclusion ........................................................................................................................................... 24

Appendix A – Available Post Event Handlers .................................................................................... 25
Introduction

Welcome to our Guide to Automating Workflows Quickly and Easily: Nintex Workflow for Project Server 2010. We will look at Nintex Workflow for Project Server 2010 and the ways it can empower your Project Managers to automate project, portfolio and collaboration workflows quickly and easily.

This guide is the second in a three-part series discussing each of the workflow types provided by Nintex Workflow for Project Server 2010 providing easy to follow step-by-step examples.

Nintex Workflow for Project Server makes the lives of Project and Portfolio Managers a lot easier, as they use a simple web-based editor to build and maintain workflows, removing the reliance on complex and costly development cycles. Nintex Workflow for Project Server is a graphical, web-based workflow designer built for the SharePoint and Project Server platforms. The key capabilities outlined in this guide are:

<table>
<thead>
<tr>
<th>Project, Portfolio, and Collaboration Workflows</th>
<th>Demand Management Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Event Driven Workflows</td>
</tr>
<tr>
<td></td>
<td>Project Workspace Workflows</td>
</tr>
<tr>
<td>Quick and Easy Functionality</td>
<td>Browser Based Designer</td>
</tr>
<tr>
<td></td>
<td>No Code Required</td>
</tr>
<tr>
<td></td>
<td>Drag and Drop</td>
</tr>
<tr>
<td>Connectivity with Other Systems</td>
<td>Query Project Data</td>
</tr>
<tr>
<td></td>
<td>Connect to Internal and External Systems</td>
</tr>
<tr>
<td></td>
<td>Integrate with Office 365</td>
</tr>
</tbody>
</table>

Figure 1 : The pillars of Nintex Workflow for Project Server

Handling the Full Range of Project, Portfolio, and Collaboration Workflows

Project and Portfolio Managers have a range of different needs for automating tasks and processes. A workflow tool should fulfil the needs for designing processes related to demand management, event driven, and collaborative workflows.

Demand Management Workflows

Demand Management Workflows allow the orchestration of an organization’s complete governance process using Project Servers demand management capabilities. Through these workflows, project ideas and projects can be guided through a governance process, whilst applying business rules, validations and approvals along the way.

Event Driven Workflows

Nintex Workflow for Project Server introduced the ability to associate workflows with Project Servers event handling infrastructure. Project Server has the ability to associate bespoke code or assemblies against events that are raised by Project Server, such as Project Published, Project Saved or Custom Field checked in etc. With Nintex Workflow for Project Server 2010, it is now possible to associate a Nintex Workflow against a subset of events within the event handler infrastructure.
quickly and easily. Nintex Workflow for Project Server Event Driven Workflows allows you to create workflows that automate scenarios such as:

- Alert Project Managers when a resource is added or activated
- Notify management when a project is published
- Query line-of-business data when a project starts
- Sync custom fields and lookup tables across Project Web App sites
- Keep team members in the loop through integration with SharePoint and much more.

![Figure 2: Project Server Service Side Event Handlers](image)

**Collaboration Workflows**

Nintex Workflow for Project Server 2010 is built on top of Nintex Workflow 2010, and provides a number of capabilities that can be leveraged within the project workspaces; ranging from document approval workflows and notifications, to stakeholders when certain risks or issues are logged, through to synchronizing content from the workspace to an external Office 365 account for collaboration with external parties.

In addition, Nintex Workflow for Project Server also leverages a number of core capabilities from Nintex Workflow 2010, including:

- Leverage pre-built workflow actions ranging from provisioning site collections, list actions, interactions with external systems such as Exchange, Lync and SQL Server directly from your workflow.
- The ability to build collaboration workflows for document approvals and other line of business applications within your SharePoint farm.
- Powerful web based drag- and- drop editing, allowing workflows to be developed in minimal time and effort.
- Powerful debugging capabilities.
- Integration with other Nintex products such as Nintex Live and Nintex Forms.
Overview of Event Handlers in Project Server 2010

Project Server provides an eventing service that allows custom code to be associated with specific events that occur in Project Server. Through these events, it is possible to encode business rules to allow Project Server to be tailored to meet an organization’s requirement. For example, in the case of actions that can occur to a project, a number of events will be raised within Project Server that a developer can choose to leverage through custom code.

The events raised by Project Server can be split into two groups:

- **Pre-events** – Events that are raised prior to the event completing. For example, when a project is published, the pre-event would be Project Publishing.
- **Post-Events** – Events that are raised once the event completes. For example, when a project is published, the post event raised would be Project Published.

With Nintex Workflow for Project Server v2.0, it is now possible to associate Nintex Workflows with Project Servers Server Side Event Handler post-events; a full list of the available event handlers that a workflow can be associated with is outlined in Appendix A.

**Why Attach Workflows to Events?**

Traditionally, leveraging the Project Server event handlers had a high barrier for use requiring the development of custom code which in turn required specialized skills to develop, test and implement. Through the use of Nintex Workflow for Project Server, it is now possible for users to encode business logic within a workflow and attach this directly to an event through a simple drag-and-drop interface.

For instance, the Project Server notification engine raises an alert of a particular format when a project is published, only allowing resources on the project to receive a notification if their assignments have changed. Through the addition of a workflow associated with the Project Published event, it is possible to automatically raise notifications to other interested parties, such as the Project Management Office, or Project Sponsor, automatically alerting them that the project has been published.

Similarly, should it be necessary to register a new project in an external system, the workflow could be extended to include a web service call to that external system, passing through relevant information about the project, directly into the external system with a minimum of development effort.
Creating an Event Driven Workflow to Automatically Backup Projects to the Archive Database

Project Server will only create archive copies of a project if the administrator decides to schedule a daily backup, or initiates a one off administrative backup. Out of the box there is no way to consistently create a backup copy of a project each time it is published.

For this example, we are going to build an event driven workflow that will initiate a backup of the project every time the schedule is published, copying the published version of the project to the archive database which can later be restored by the administrator if required.

Building the archive workflow

1. To create an Event Driven Workflow, select Site Actions > Project Server Workflow > Manage Event Driven Workflows.

Event Driven Workflows are different from Demand Management Workflows and have a number of different actions and options to choose from. To create a workflow, click on the New option of the Manage Event Driven Workflows Ribbon.
2. To create a new Event Driven Workflow, select **New**.

Event Driven Workflows have more in common with traditional SharePoint workflows than Demand management workflows, and as a result a majority of the Project Server Demand Management actions are not available. However, there are three core Project Server actions which can be leveraged.

![Event Driven Workflow Project Server Actions](image)

**Figure 5: Event Driven Workflow Project Server Actions**

**Retrieve event data** – This action is specific to Event Driven Workflows and allows the context of the event to be passed to the workflow. For instance, if the workflow is going to run on a Project Published Event, the Retrieve Event Data Action will be configured to listen for the Project Published event and can be further configured to pass context information such as the Project Name and ProjectUID which can be used in the workflow.

**Query Project Server** – This action allows data within Project Server to be queried using Project Server’s API, the Project Server Interface or PSI. The Query Project Server allows any Project Server PSI methods to be called and the datasets returned filtered and leveraged in workflows.

**Update Project Properties** – This action allows attributes of the project to be updated through the PSI. This action is similar to the Update Project Property Demand Management Action, but works outside of the demand management infrastructure and allows access to non-demand management properties, such as the Project Owner.
3. Drag a **Retrieve event data** action on to the design canvas.

![Figure 6: Retrieve Event Data](image)

4. Double-click on the action and configure the action to listen for the Project Published event as outlined in **Figure 7: Configure Retrieve Event Data Action**.

5. a) In the **Event** field, enter `Project; Published`.

![Figure 7: Configure Retrieve Event Data Action](image)

b) In the **Data to retrieve** drop-down menu, choose `ProjectGUID` and click **Add**.

For this workflow we need to put the `ProjectGUID` into a workflow variable so we can tell the archive process which project to backup.

1. To create a workflow variable, select **Variables** from the Ribbon and click on **New**.
2. In the **Name** field, type “ProjectUID” and select **Single line of text** as the **Type**.

![Create Workflow Variable](image)

**Figure 8 : Create ProjectUID Workflow Variable**

3. Click on **Save**.
4. In the **ProjectGUID** field, select the newly created variable. The configured dialog should appear as below.

![Configure Action - Retrieve event data](image)

**Figure 9 : Completed Retrieve Event Data Workflow Action**

5. To complete the configuration of the **Retrieve event data** action, click on **Save**.
In order to perform the project backup to the Archive Database, it is necessary to call one of the PSI methods using a web service. Nintex Workflow includes a **Call web service** action that can be leveraged within our event driven workflow.

6. In the Workflow Actions toolbox, select the **Call web service** action and drag it onto the design canvas. The workflow should look as below.

![Call Web Service Workflow Action](image)

7. Double-click on the **Call Web Service** action to configure it.
In the URL field we need to enter the URL of the archive web service we are going to use. In this example, the URL is http://project.contoso.com/PWA/_vti_bin/PSI/archive.asmx?wsdl. The URL in your environment may have a different domain and PWA instance name, but the _vti_bin/PSI/archive.asmx?wsdl should remain the same.

The web service will require authentication, so enter credentials of a user that has administrator access (required to initiate the archive). It may be a good idea to create a special administrator login which only has permissions to backup for use in this action.

![Configure Action - Call web service](image)

Figure 11: Configure the Call Web Service Action
8. Once the correct credentials have been entered, click on the **Refresh** button. This will cause the action to attempt to connect to the web service. If all is successful the list of Web Methods will be populated with all the methods available.

![Figure 12: Available Web Methods](image)

---

**Back to Contents**
For the purposes of this workflow, it is necessary to use the **QueueArchiveProject** method. This method will use the Project Server queue to archive our project. Once the method has been selected, the action configuration box will update to show the required parameters for the chosen method.

![Figure 13: Web Service Parameters](image)

As you can see from the above, the **QueueArchiveProject** method requires six parameters:

- **jobUID** – This is a GUID that is required for the Project Server queuing mechanism. In this example, a new GUID will be generated using the fn-NewGuid() inline function of Nintex Workflow.
- **projectUID** – This is the project UID of the Project to be archived, in this example, this is contained in the variable ProjectUID created above.
- **archivedProjectUID** – a new GUID that will be assigned to the archived project. Again, in this example the fn-NewGuid() inline function will be used.
- **versionDescription** – a string that will be stored against the archived project that is viewable in the Administrative Restore function.
- **retentionPolicy** – an integer that will determine how the project is retained. For this example, the value 0 will be used, meaning the backup is ignored by the retention policy.
- **permanentArchive** – a Boolean that allows the project backup to be backed up permanently and ignore the retentionPolicy parameter. In this example, this is set to false.
Note: Full details of each of these methods and their associated parameters can be found on MSDN at http://msdn.microsoft.com/en-us/library/ee767707.aspx.


Once the values for the Web Service input have been entered, the configuration screen should look like the image below.

![Figure 14: Partially Completed Web Service Parameters](image)

You will notice that the versionDescription parameter is not configured. It would be possible to simply put some text in the field stating the project was archived, however as we may be creating multiple backups each day and Project Server only records the day a backup was taken, the versionDescription field will be used to record the date and time the backup was taken. To achieve this, click Save to close the dialog and add another action to the workflow.
1. Drag a **Build string** action onto the design canvas, between the two existing actions. The workflow should look like **Figure 15: Build String Added to Workflow**.

![Figure 15: Build String Added to Workflow](image)

2. Double-click on the **Build string** action to open the configuration dialog.
3. Click on the **Variables** button in the Ribbon to create a new string variable called **ArchiveString** to store our description into.
   - In the **Name** field, type “ArchiveString” and select **Single line of text** as the **Type**.
4. In the configuration dialog, click on **Insert Reference** to add the current date and time to the string.

5. Type “Project Publish Archive”.

6. Choose to store the result in the **ArchiveString** variable created above.
Lastly, all that is required in the workflow is to open the Call web service action configuration again and associate the ArchiveString variable with the versionDescription parameter. Once complete the dialog should look as below.

![Completed Call Web Service Action](image)

**Figure 18 : Completed Call Web Service Action**

7. Choose **Save**.
8. Click the **Publish** button in the Ribbon to publish the workflow.
When naming the workflow, it is a good idea to make sure you give it a meaningful name that describes not only the workflow functionality, but also the Event that the retrieve event data in the first action is listening for, similar to the name seen in Figure 19: Save and Publish Workflow.

Once the workflow has been published it will be available to associate with the event in Project Server.
Associating the workflow with an event

Before an event driven workflow can be used, it is necessary to associate with the event in Project Server. Nintex Workflow takes some of the pain out of this process by providing a handy interface to perform the association.

1. To associate our workflow created above, in the Manage Event Driven Workflows tab, select Create.

![Create an Event Association Menu](image)

Figure 20: Create an Event Association Menu

A dialog will be displayed allowing you to associate the workflow with the Project Server event.

![Associate a Workflow to an Event](image)

Figure 21: Associate a Workflow to an Event

2. In the Event field, select Project; Published.

*Note: It is important these values are the same as configured in the Retrieve event data action, otherwise your workflow may not receive the data it is expecting and fail.*
3. In the **Workflow** field, select the workflow we published above (if your workflow is not listed, check the workflow is published).

4. Click on **Save**.

Note: The process of creating the event is performed by a timer job behind the scenes and can take a few moments to complete.

To check that the event association is successful, navigate to Project Server’s **Server Settings > Server Side Event Handlers** and click on the event you have associated to the workflow. If the event has been associated, you should see Nintex Workflow Event Handler line in the event handler section.

![Server Side Event Handlers Screen in Project Server](image)

*Figure 22: Server Side Event Handlers Screen in Project Server*
Running the Workflow
Once the workflow has been built and associated with an event, you may test and run the workflow. In this example, to test the workflow we will publish a project, either from the PWA interface or directly from Project Professional. We are going to publish the project Automated Software Design Architecture.

Figure 23: Publish the Project from Project Professional
Once the project has been published, the event will fire and call the workflow. This will create a backup of the project in the Archive database. The easiest way to see if the workflow worked correctly is to open Project Server. Navigate to Server Settings > Administrative Restore and look at the various projects available to restore.

Figure 24: Administrative Restore Screen
As you can see, the Automated Software Design Architecture Upgrade project has been backed up automatically, and has a version description comment from our workflow.
Viewing the status of a workflow

In addition, like all Nintex Workflows, it is possible to view the status of all workflows that are in progress, completed, cancelled or in error and to then drill down and see where that error occurred in a graphical interface.

1. Navigate to the Manage Event Driven Workflows Ribbon, and select Workflow History.

The Workflow History will show the status of the workflows, in this case ours completed successfully.
2. Clicking on the workflow name will allow you to drill down further and view the status of each workflow action, including data on when the actions started and completed.

![Workflow Progress Screen](image-url)

Figure 27: Workflow Progress Screen
Conclusion

Nintex Workflow for Project Server 2010 empowers project and portfolio managers to design and implement Demand Management Workflows without the need for complex and costly development cycles.

In part three of this series of product guides, we will show how the underlying Nintex Workflow integration with SharePoint helps improve collaboration and communications across the project team during project execution and management.

Nintex Workflow for Project Server 2010 helps you to execute your projects on time, within budget, aligned to your corporate strategies.

Download a free 30 Day trial now at: www.nintex.com/project.
## Appendix A – Post Event Handlers

<table>
<thead>
<tr>
<th>Type</th>
<th>Event</th>
<th>Type</th>
<th>Event</th>
<th>Type</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td>AdminAdSyncERPSynchronized</td>
<td>Project</td>
<td>ProjectActivityUpgraded</td>
<td>Reporting cont.</td>
<td>ReportingResourceCapacitiesChanged</td>
</tr>
<tr>
<td>Admin</td>
<td>AdminAdSyncGroupSynchronized</td>
<td></td>
<td>ProjectAdded</td>
<td></td>
<td>ReportingResourceCapacityTimeRangeChanged</td>
</tr>
<tr>
<td>Admin</td>
<td>AdminAdSyncGroupSynchronized</td>
<td></td>
<td>ProjectCheckIn</td>
<td></td>
<td>ReportingResourceChanged</td>
</tr>
<tr>
<td>Admin</td>
<td>AdminLineClassUpdated</td>
<td></td>
<td>ProjectCreated</td>
<td></td>
<td>ReportingResourceCreated</td>
</tr>
<tr>
<td>Admin</td>
<td>AdminReportingPeriodUpdated</td>
<td></td>
<td>ProjectDeleted</td>
<td></td>
<td>ReportingResourceDeleted</td>
</tr>
<tr>
<td>Admin</td>
<td>AdminStatusReportsDeleted</td>
<td></td>
<td>ProjectEntitesDeleted</td>
<td></td>
<td>ReportingTimesheetAdjusted</td>
</tr>
<tr>
<td>Calendar</td>
<td>CalendarChanged</td>
<td></td>
<td>ProjectPublished</td>
<td>Reporting cont.</td>
<td>ReportingResourceCapacitiesChanged</td>
</tr>
<tr>
<td>Calendar</td>
<td>CalendarCheckedIn</td>
<td></td>
<td>ProjectSaved</td>
<td>ReportingTimesheetDeleted</td>
<td>ReportingResourceCapacityTimeRangeChanged</td>
</tr>
<tr>
<td>Calendar</td>
<td>CalendarCheckedOut</td>
<td></td>
<td>ProjectSaveFailed</td>
<td>ReportingTimesheetPeriodChanged</td>
<td>ReportingResourceChanged</td>
</tr>
<tr>
<td>Calendar</td>
<td>CalendarCreated</td>
<td></td>
<td>ProjectUpdated</td>
<td>ReportingTimesheetPeriodCreated</td>
<td></td>
</tr>
<tr>
<td>Calendar</td>
<td>CalendarDeleted</td>
<td></td>
<td>ProjectWssWorkspaceAddressDeleted</td>
<td>ReportingTimesheetPeriodDeleted</td>
<td></td>
</tr>
<tr>
<td>CubeAdmin</td>
<td>CubeAdminCubeBuilt</td>
<td>Reporting</td>
<td>ReportingCustomFieldChanged</td>
<td></td>
<td>ReportingTimesheetSaved</td>
</tr>
<tr>
<td>CubeAdmin</td>
<td>CubeAdminCubeProcessed</td>
<td></td>
<td>ReportingCustomFieldDeleted</td>
<td></td>
<td>ReportingTimesheetStatusChanged</td>
</tr>
<tr>
<td>CustomFields</td>
<td>CustomFieldsCheckedIn</td>
<td></td>
<td>ReportingFiscalPeriodChanged</td>
<td>Resource</td>
<td>ResourceActivated</td>
</tr>
<tr>
<td>CustomFields</td>
<td>CustomFieldsCheckedOut</td>
<td></td>
<td>ReportingFiscalPeriodDeleted</td>
<td>Resource</td>
<td>ResourceChanged</td>
</tr>
<tr>
<td>CustomFields</td>
<td>CustomFieldsCreated</td>
<td></td>
<td>ReportingLookupTableChanged</td>
<td>Resource</td>
<td>ResourceCheckedIn</td>
</tr>
<tr>
<td>CustomFields</td>
<td>CustomFieldsDeleted</td>
<td></td>
<td>ReportingLookupTableDeleted</td>
<td>Resource</td>
<td>ResourceCheckedOut</td>
</tr>
<tr>
<td>LookupTable</td>
<td>LookupTableCheckedIn</td>
<td></td>
<td>ReportingProjectChanged</td>
<td>Resource</td>
<td>ResourceDeactivated</td>
</tr>
<tr>
<td>LookupTable</td>
<td>LookupTableCheckedOut</td>
<td></td>
<td>ReportingProjectDeleted</td>
<td>Resource</td>
<td>ResourceDeleted</td>
</tr>
<tr>
<td>LookupTable</td>
<td>LookupTableCreated</td>
<td></td>
<td>ReportingProjectCreated</td>
<td>Resource</td>
<td>ResourceSetAuthorization</td>
</tr>
<tr>
<td>LookupTable</td>
<td>LookupTableDeleted</td>
<td></td>
<td>ReportingProjectWorkspaceChanged</td>
<td>Resource</td>
<td>ResourcePlanPublished</td>
</tr>
<tr>
<td>Notifications</td>
<td>NotificationsSent</td>
<td></td>
<td>ReportingProjectWorkspaceCreated</td>
<td>Security</td>
<td>SecurityCategoryCreated</td>
</tr>
<tr>
<td>Optimizer</td>
<td>OptimizerSolutionCommitted</td>
<td></td>
<td>ReportingProjectWorkspaceDeleted</td>
<td>Security</td>
<td>SecurityCategoryDeleted</td>
</tr>
<tr>
<td>Optimizer</td>
<td>PlannerSolutionCommitted</td>
<td></td>
<td>ReportingProjectWorkspaceDeleted</td>
<td>Security</td>
<td>SecurityCategoryUpdated</td>
</tr>
<tr>
<td>Type</td>
<td>Event</td>
<td>Type</td>
<td>Event</td>
<td>Type</td>
<td>Event</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>SecurityGroupDeleted</td>
<td></td>
<td>StatusingStatusUpdated</td>
<td></td>
<td>User Delegation</td>
</tr>
<tr>
<td></td>
<td>SecurityGroupUpdated</td>
<td></td>
<td>StatusingTaskCreated</td>
<td></td>
<td>UserDelegationActivated</td>
</tr>
<tr>
<td></td>
<td>SecurityOrganizationalPermissionsUpdated</td>
<td></td>
<td>StatusReportsRequestCreated</td>
<td></td>
<td>UserDelegationChanged</td>
</tr>
<tr>
<td></td>
<td>SecurityTemplateCreated</td>
<td></td>
<td>StatusReportsRequestUpdated</td>
<td></td>
<td>UserDelegationCreated</td>
</tr>
<tr>
<td></td>
<td>SecurityTemplateDeleted</td>
<td></td>
<td>StatusReportsResponseCreated</td>
<td></td>
<td>UserDelegationDeactivated</td>
</tr>
<tr>
<td></td>
<td>SecurityTemplateUpdated</td>
<td></td>
<td>StatusReportsResponseUpdated</td>
<td></td>
<td>UserDelegationDeleted</td>
</tr>
<tr>
<td>Statusing</td>
<td>StatusingApplied</td>
<td>Timesheets</td>
<td>TimesheetCreated</td>
<td>Workflow</td>
<td>WorkflowCompleted</td>
</tr>
<tr>
<td></td>
<td>StatusingApprovalsUpdated</td>
<td></td>
<td>TimesheetDeleted</td>
<td></td>
<td>WorkflowProjectTypeChanged</td>
</tr>
<tr>
<td></td>
<td>StatusingAssignmentDelegated</td>
<td></td>
<td>TimesheetLineApproved</td>
<td></td>
<td>WorkflowStageEntered</td>
</tr>
<tr>
<td></td>
<td>StatusingAssignmentDeleted</td>
<td></td>
<td>TimesheetRecalled</td>
<td></td>
<td>WorkflowStarted</td>
</tr>
<tr>
<td></td>
<td>StatusingAssignmentWorkDataSet</td>
<td></td>
<td>TimesheetReviewed</td>
<td>WSSinterop</td>
<td>WssInteropWssWorkspaceCreated</td>
</tr>
<tr>
<td></td>
<td>StatusingHistoryDeleted</td>
<td></td>
<td>TimesheetSubmitted</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>