***Getting started with TFS Event Workflows***

* **How to install the TFS Event Workflows on TFS 2012, TFS 2013 and TFS 2015 (version >= 0.9.\*)**

Starting with version 0.9.41488.0 TfsEventWorkflows no longer support TFS 2010. Also there’s currently no setup routine or installer package.

* **Manual Installation of TFS event workflows for synchronous execution**

Download TFS Event Workflows from [http://tfseventworkflows.codeplex.com](http://tfseventworkflows.codeplex.com/).

TFS 2012: download the file TfsEventWorkflows2012-<update>-<version>.zip

TFS 2013: download the file TfsEventWorkflows2013-<update>-<version>.zip

TFS 2015: download the file TfsEventWorkflows2015-<update>-<version>.zip

**Attention:** Please read the notes regarding TFS Updates on the TFS Event Workflows Web-Site. You should use the correct version for your installed update.

**Extract the files** in the ZIP to a local folder <tewfiles>.

**Attention**: Please ensure the files are not “blocked” after unzip, TFS does not load “blocked” files. You can use one the following PowerShell commands to “unblock” all the extracted files at once. Open PowerShell and set directoy to the location whre you have extracted the files to.

* Get-ChildItem -Recurse -File | Unblock-File
* dir -Recurse -File | Unblock-File

**Copy** the following files:

* <tewfiles>*\Plugins\artiso.TfsEventWorkflows.LoggingLib.dll*
* <tewfiles>*\Plugins\log4net.dll*
* <tewfiles*>\Plugins\artiso.TfsEventWorkflows.TfsEventWorkflowsServerPlugin.dll*
* <tewfiles>*\Plugins\artiso.TfsEventWorkflows.TfsEventWorkflowsServerPlugin.dll.config*

to the <web service plugin directory> of you TFS application tier installation. The default installations paths are:

* **TFS 2012**: *C:\Program Files\Microsoft Team Foundation Server 11.0\Application Tier\Web Services\bin\Plugins*
* **TFS 2013:** *C:\Program Files\Microsoft Team Foundation Server 12.0\Application Tier\Web Services\bin\Plugins*
* **TFS 2015:** *C:\Program Files\Microsoft Team Foundation Server 14.0\Application Tier\Web Services\bin\Plugins*

**Adjust the logging path** in the config file *artiso.TfsEventWorkflows.TfsEventWorkflowsServerPlugin.dll.config*

<log4net>

<appender name=" TfsEventWorkflow-WebServiceAppender" type="log4net.Appender.FileAppender">

<!— skipped -->

<file type="log4net.Util.PatternString"

value="**C:\data\temp\tfseventworkflows\**tew-%date{yyyy-MM-dd}-WebService.log"/>

<!— skipped -->

</appender>

<!— skipped -->

</log4net>

**Attention:** Ensure that your TFS service account has read/write permissions to the logging path!

Test whether the logging is configured correctly. To do this use Visual Studio to connect to your TFS server and change a workitem. A log file should be created in the logging directory.

Next setup and test a simple workflow. To do this copy the following files to your <web service plugin directory>.

* <tewfiles>\WorkflowActivities\artiso.TFSEventWorkflows.TFSActivitiesLib.dll
* <tewfiles>\WorkflowSamples\EmptyWorkflow.xaml

Activate the workflow in the config file artiso.TfsEventWorkflows.TfsEventWorkflowsServerPlugin.dll.config

<tfsEventConfig>

<tfsEvents>

<add name="workItemChangedEvent1"

fullTypeName="Microsoft.TeamFoundation.WorkItemTracking.Server.WorkItemChangedEvent"

eventAssemblyName="Microsoft.TeamFoundation.WorkItemTracking.Server.Dataaccesslayer"

workflowFileName="EmptyWorkflow.xaml"/>

<!— skipped -->

</tfsEvents>

</tfsEventConfig>

Use Visual Studio to change a workitem. Now you should get additional information in the logfile about execution of this workflow

**Note**: Microsoft has changed the packaging of classes in the TFS server object model slightly from version to version. This also had happened for some of the events, so please verify whether the provided assembly name matches for the TFS version you use. In the .config sample we have added comments regarding the common events affected.

If logging does not function so far do the following

* Check your TFS application tier service account in the TFS Administration Console
* Ensure that the service account has access to the configured logging directory
* Restart IIS by using iisrest.exe

Logging should work at this point.

Finally test a workflow accessing workitem data. Copy the following files to your <web service plugin directory>.

* <tewfiles>\WorkflowSamples\DumpWorkitem.xaml

Activate the workflow in the config file artiso.TfsEventWorkflows.TfsEventWorkflowsServerPlugin.dll.config

<tfsEventConfig>

<tfsEvents>

<!— skipped -->

<add name="workItemChangedEvent2"

fullTypeName="Microsoft.TeamFoundation.WorkItemTracking.Server.WorkItemChangedEvent"

eventAssemblyName="Microsoft.TeamFoundation.WorkItemTracking.Server.Dataaccesslayer"

workflowFileName="DumpWorkitem.xaml"/>

<!— skipped -->

</tfsEvents>

</tfsEventConfig>

Use Visual Studio to change a workitem. Now you should get additional information in the logfile about the workitem changed. If you get an error in this step do the following

* Ensure that the service account has client access to the TFS app tier. To do this login using the service account and start Visual Studio to Connect to the server.
* If this fails first grant access for your service account
* **Manual Installation of TFS event workflows for asynchronous execution**

The basic steps are the same as for synchronous execution. So repeat the steps, but use the following <job agent plugin directory> as the target directory.

* *TFS 2012: C:\Program Files\Microsoft Team Foundation Server 11.0\Application Tier\ TFSJobAgent\plugins*
* *TFS 2013: C:\Program Files\Microsoft Team Foundation Server 12.0\Application Tier\ TFSJobAgent\plugins*
* *TFS 2015: C:\Program Files\Microsoft Team Foundation Server 12.0\Application Tier\ TFSJobAgent\plugins*

You can skip the copying of the workflow xaml files and the activation of the workflows in the config. This is done in the <web services plugin directory> for asynchronous execution also.

Attention: you still have to configure the logging directory in workflow in the config file artiso.TfsEventWorkflows.TfsEventWorkflowsServerPlugin.dll.config residing in the <job agent plugin directory>. Ensure to use different log files for web services and job agent.

<log4net>

<appender name=" TfsEventWorkflow-**JobAgentAppender**" type="log4net.Appender.FileAppender">

<!— skipped -->

<file type="log4net.Util.PatternString"

value="**C:\data\temp\tfseventworkflows\**tew-%date{yyyy-MM-dd}-**JobAgent**.log"/>

<!— skipped -->

</appender>

<root>

<!— skipped -->

<appender-ref ref="TfsEventWorkflow-**JobAgentAppender**"/>

</root>

</log4net>

Now restart the service “Visual Studio Background Job Agent” using the services control panel applet.

To activate one of the workflows go to the <**web service** plugin directory> and edit artiso.TfsEventWorkflows.TfsEventWorkflowsServerPlugin.dll.config there

<tfsEventConfig>

<tfsEvents>

<add name="workItemChangedEvent1"

fullTypeName="Microsoft.TeamFoundation.WorkItemTracking.Server.WorkItemChangedEvent"

eventAssemblyName="Microsoft.TeamFoundation.WorkItemTracking.Server.Dataaccesslayer"

workflowFileName="EmptyWorkflow.xaml"

**async="true"/>**

<add name="workItemChangedEvent2"

fullTypeName="Microsoft.TeamFoundation.WorkItemTracking.Server.WorkItemChangedEvent"

eventAssemblyName="Microsoft.TeamFoundation.WorkItemTracking.Server.Dataaccesslayer"

workflowFileName="DumpWorkitem.xaml"

**async="true"/>**

<!— skipped -->

</tfsEvents>

</tfsEventConfig>

Start Visual Studio, change a workitem and verify the log.

**Hint on asynchronous operations at all**: The TFS web service can timeout. So if you configure larger workflows and trigger a lot of operations simultaneously (e.g. bulk edit of workitems in Excel), the execution can timeout. To avoid this long term operations should be transferred to the Job Agent. That’s exactly what happens when you use the async keywork in the config file.

* **How to create a custom workflow in Visual Studio**

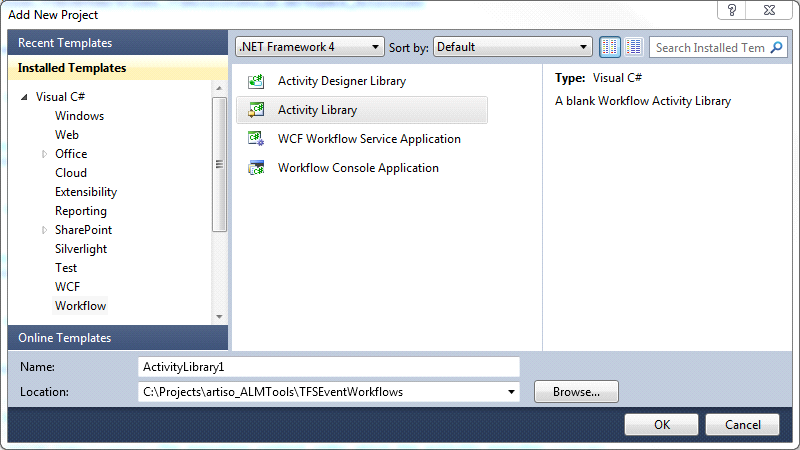
This sample shows how to execute a workflow when a work item is saved.

First we create a workflow which handles the WorkItemChangedEvent. The workflow will add a new work item of the type task to a new user story. So each time a new user story is created by a user a new task will be added as child.

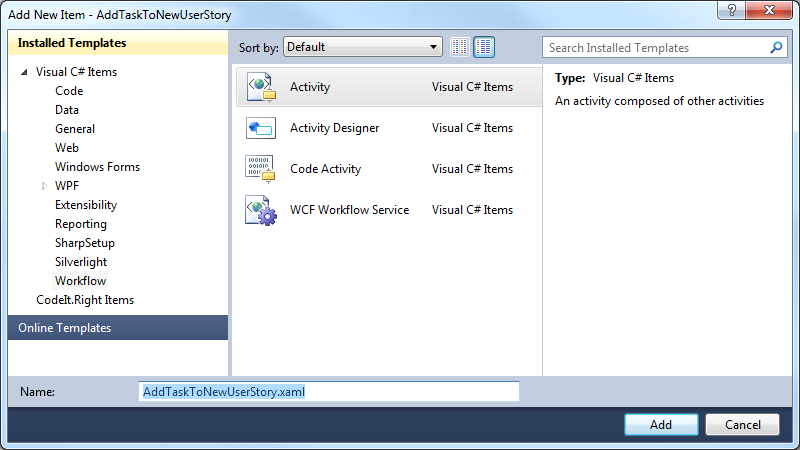
* **Creating a custom workflow with the TFSActivitiesLib**

First step is to create the workflow.

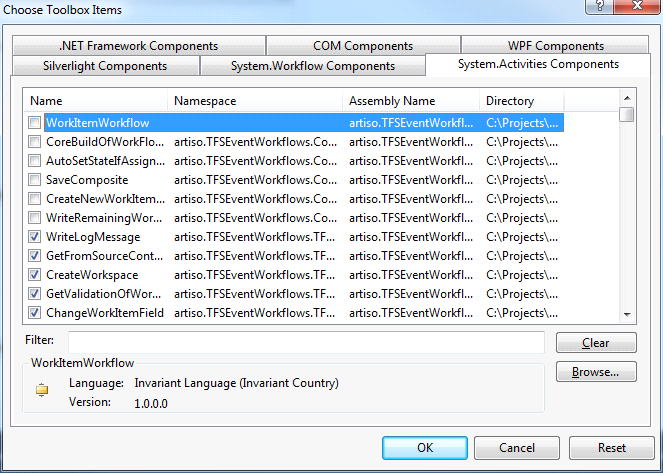
* Create a new solution. Add a new "Activity Library" project from the "Workflow" template.



* Add a new workflow activity with the name AddTaskToNewUserStory.xaml to the project.



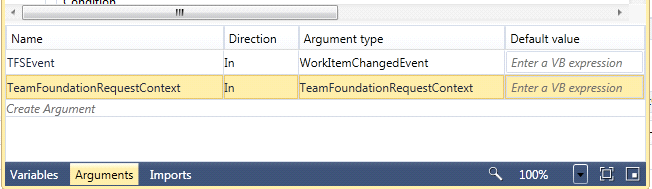
* Add a reference to the TFSActivitiesLib (artiso.TFSEventWorkflows.TFSActivitiesLib.dll).
* Open the AddTaskToNewUserStory.xaml editor and add the TFS Event Workflows activities to the toolbox. Right-click on the Toolbox and select "Add Tab". Type in a name for the tab like "TFSEventWorkflows". Right-click on the new tab and select "Choose Items…".



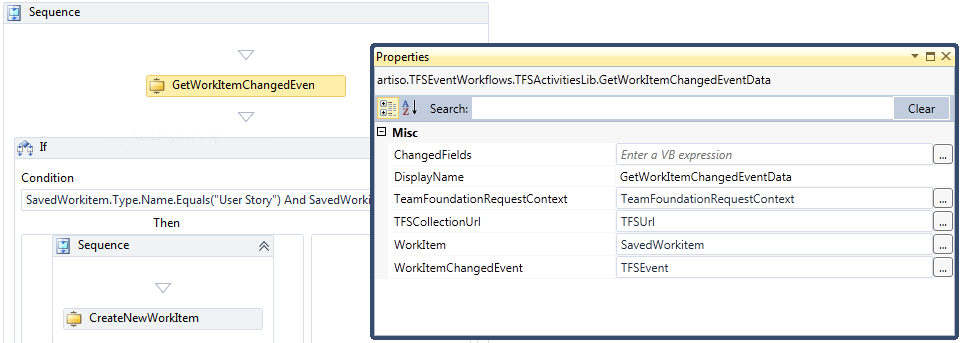
* In the "Choose Toolbox Items" window click "Browse…" and select the artiso.TFSEventWorkflows.TFSActivitiesLib.dll.
* The activities from the TFSActivitiesLib are in the Toolbox and can be dragged into the workflow designer.

Now we can design the workflow.

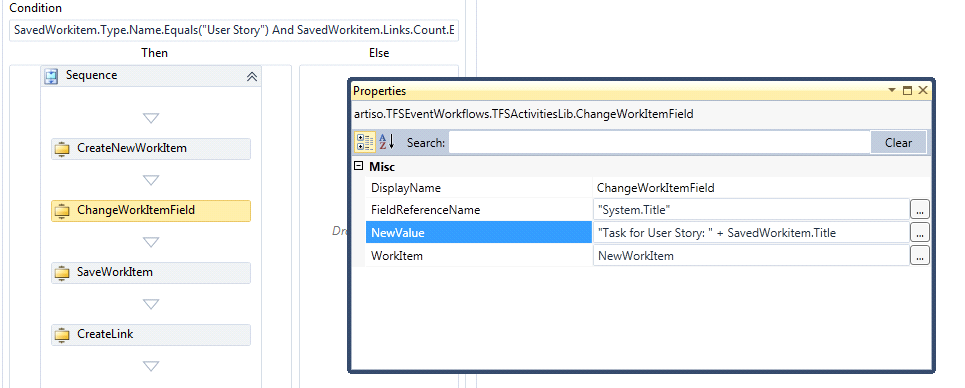
* First we need to define the InArguments for the workflow. A WorkItemChangedEvent workflow takes two arguments, the WorkItemChangedEvent and TeamFoundationRequestContext.   
  Add a reference to:  
  Microsoft.TeamFoundation.WorkItemTracking.Server.Dataaccesslayer.dll  
  Microsoft.TeamFoundation.Framework.Server.dll  
  Microsoft.TeamFoundation.WorkItemTracking.Client



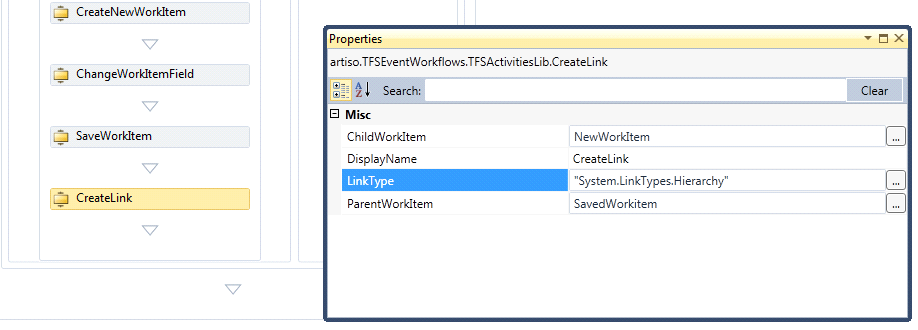
* To process the InArguments we drag the activity GetWorkitemChangedEventData to the workflow.  
  The activity takes the InArguments and returns the saved work item, the TFS collection URL and the changed fields of the work item.



* After we get the saved workitem from the GetWorkitemChangedEventData activity we check if the work item type is a "User Story" and if the revision of the work item is 1, which means the work item is new.  
  SavedWorkitem.Type.Name.Equals("User Story") And SavedWorkitem.Revision.Equals(1)
* If the condition is true we create a new work item with the CreateNewWorkItem activity, change the title of the new work item with the ChangeWorkItemField activity and save the new work item.



* After the save we create a hierarchy link between the user story (parent) and the new task (child) with the CreateLink activity.



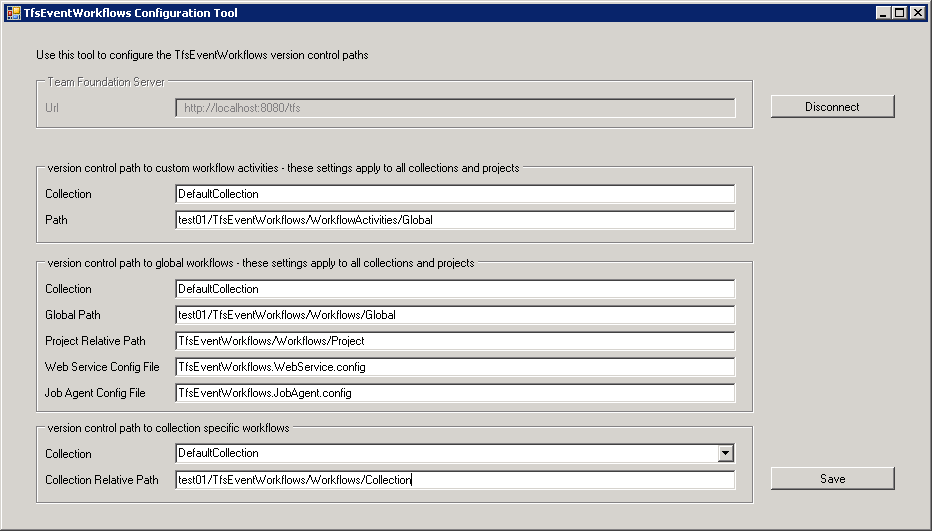
* **Configure TFS Event Workflows to store Workflows, Configurations and binary Workflow Activities in the Version Control**

Starting with version 0.9.41488.0 TfsEventWorkflows can store the following information in the Version Control

* Workflows – can be stored global, collection specific or project specific
* Configuration files – can be stored global, collection specific or project specific
* Custom Activities – can be stored global

This allows you to store and activate all workflows using a checkin to the version control, no copy to application tier folders is necessary at all. This makes the configuration of TfsEventWorksflows much smoother, especially in multiple app tier environments.

You can configure the version control paths using the TfsEventWorkflowsConfiguration.exe tool



|  |  |  |
| --- | --- | --- |
| **Group** | Value | Description |
| **version control path to custom workflow activities** | Collection | Name of the Team Project Collection containing the folder for all Custom Workflow Activities |
|  | Path | Absolute Path of the version control folder |
| **version control path to global workflows** | Collection | Name of the Team Project Collection containing the folder for all Global Workflows |
|  | Path | Absolute Path of the version control folder |
|  | Project Relative Path | Relative Path of the version control folder containing Team Project specific workflow. This path is automatically appended to the Team Project folder in version control |
|  | Web Service Config File | Name of the configuration file used by the TFS Web Service Pluggin. The file is fetched from the same version control folder like the workflow files. |
|  | Job Agent Config File | DEPRECATED  Name of the configuration file used by the TFS Job Agent Pluggin. The file is fetched from the same version control folder like the workflow files. Use the new async Attribute instead |
| **version control path to collection specific workflows** | Collection | Choose one from the drop down |
|  | Collection Relative Path | Absolute Path of the version control folder |

|  |  |  |
| --- | --- | --- |
| **Attribute in the config file** | Allowed on | Description |
| **name** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder  Project version control folder | Mandatory, must be unique in the config file |
| **fullTypeName** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder  Project version control folder | Mandatory, the fully qualified type name of the event |
| **eventAssemblyName** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder  Project version control folder | Mandatory, assembly name defining the type of the event |
| **workflowFileName** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder  Project version control folder | Mandatory, name of the workflow file used to handle the event. |
| **collection** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder | Optional (default: <empty>)  collection to apply this event workflow, <empty> means all collections  ignored, if the config already resides in collection or project specific version control path |
| **project** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder | Optional (default: <empty>)  project to apply this event workflow, <empty> means all projects  ignored, if the config already resides in project specific version control path |
| **disabled** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder  Project version control folder | Optional (default: false)  Simple option to disable a workflow temporary |
| **async** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder  Project version control folder | Optional (default: false)  If true the event is automatically queued to the Job Agent.  the QueueToJobAgent activity and sample workflow are DEPRECTAED now, new config should use async instead |
| **trace** | Web Service Plugin Directory  Job Agent Plugin Directory  Global version control folder  Collection version control folder  Project version control folder | Optional (default: false)  If true the workflow execution is traced to the log file. |

**Some Notes an async and QueueToJobAgent activity and sample workflow**

The new async attribute allows you to queue a workflow very simple to the TFS Job Agent and run it thus in an async manner. Because this can be configured directly in the config file used by the Web Service a config file for the Job Agent and the use of the QueueToJobAgent activity and sample workflow gets obsolete at all. So they are deprecated and only provided for compatibility with older version of TfsEventWorkflows to simplify the upgrade process.

**Some Notes to custom activity assemblies**

Storing these files in the version control simplifies their distribution. The assemblies will be get from version control and load by the TfsEventWorkflows engine on each app tier on demand. Hint: once the assembly is load from a workflow the file is locked by the Job Agent or the Web Service and update is blocked until the Plugin is reloaded. This happens after the TFS App Pool timeout or after restarting the TFS services.

* **How to install the TFS Event Workflows on TFS 2010 (version <= 0.8.\*)**
* Open the TFS Event Workflows project, build the solution and copy the assemblies from the output to the plugins folder of the TFS.

Path to TFS plugins directory:

…\Microsoft Team Foundation Server 2010\Application Tier\Web Services\bin\Plugins

Required assemblies:

* artiso.TFSEventWorkflows.LoggingLib.dll
* artiso.TFSEventWorkflows.TFSActivitiesLib.dll
* artiso.TFSEventWorkflows.TFSEventWorkflowsServerPlugin.dll
* artiso.TFSEventWorkflows.TFSEventWorkflowsServerPlugin.dll.config
* log4net.dll
* MyWorkflowFile.xaml

**TFS Event Workflows Setup:**

The TFS Event Workflow can also be installed by executing the "TFSEventWorkflowsSetup.msi" located in the TFSEventWorkflowsSetup project. Set the installation path to the plugins folder of the TFS.

(…\Microsoft Team Foundation Server 2010\Application Tier\Web Services\bin\Plugins).

* Copy the workflow file AddTaskToNewUserStory.xaml to the plugins directory.
* To subscribe to the WorkItemChangedEvent must be configured in the configuration file (artiso.TFSEventWorkflows.TFSEventWorkflowsServerPlugin.dll.config).

In the tfsEventConfig section the following values must be set:

**name:**

A name for the entry. Must be unique.

**fullTypeName:**

The full name of the TFS event.

**eventAssemblyName:**

The full name of the assembly which contains the TFS event.

**workflowFileName:**

The path of the workflow file which will be executed with the event. If the workflow file is in the same directory as the server plugin dll the workflow file name is sufficient (e.g. MyWorkflowFile.xaml).

<tfsEventConfig>

<tfsEvents>

<add name="workItemChangedEvent1"

fullTypeName="Microsoft.TeamFoundation.WorkItemTracking.Server.WorkItemChangedEvent"

eventAssemblyName="Microsoft.TeamFoundation.WorkItemTracking.Server.Dataaccesslayer"

workflowFileName="MyWorkflowFile.xaml" />

</tfsEvents>

</tfsEventConfig>

* Save the configuration file and create a new user story.