AIT - WordToTFS User Guide

V6.3.0.0

Contents
System Requirements .................................................................................................................. 3
Quick Start: Installation ........................................................................................................... 3
Quick Start: Overview .............................................................................................................. 3
Quick Start: Add a Template Location ..................................................................................... 5
Quick Start: Connect, Create, Publish ................................................................................... 6
Product Usage .......................................................................................................................... 7
   Connect to a Team Project ..................................................................................................... 7
   Choose a Template ................................................................................................................ 7
   Manage Available Templates ............................................................................................... 8
   Insert Work Items ................................................................................................................ 9
   Support for any kind of work item ........................................................................................ 10
   Work Item Overview ............................................................................................................ 10
   Publish work items to TFS .................................................................................................. 10
   Edit default values for the current template ........................................................................ 12
   Delete work item ids from an existing document ................................................................. 12
   Import Existing Work Items from TFS ................................................................................. 13
   Insert Area and Iteration Path Values ............................................................................... 14
   Refresh all or selected work items ....................................................................................... 15
   Using Headers ...................................................................................................................... 15
   Provide Feedback .................................................................................................................. 15
   Test Management Reports .................................................................................................. 16
   Structure of generated report ............................................................................................... 22
   Enhanced Document Structure ............................................................................................ 25
   Managing Templates ............................................................................................................ 26
   Add a Template Location ..................................................................................................... 26
   Customizing the template configuration: w2t file ............................................................... 28
   Customizing the buttons of the ribbon .................................................................................. 30
   Creating a new work item mapping ...................................................................................... 31
   Defining Headers .................................................................................................................. 42
   Defining Variables ............................................................................................................... 45
   Collapsing/Expanding of the query tree .............................................................................. 45
Customizing the template layout: .xml files................................................................. 45
Configuring the test reporting extension ........................................................................ 45
  General test configuration ......................................................................................... 46
  Test specification report configuration ..................................................................... 46
  Test result report configuration ............................................................................... 47
  Test run result ......................................................................................................... 49
Default values for test reporting .................................................................................. 50
Identify test suite selection by tree path ...................................................................... 52
Templates .................................................................................................................. 52
Replacement with variables within templates............................................................. 56
Invoke methods from within the template .................................................................. 56
Object used to resolve property value ........................................................................ 56
Object model extensions ............................................................................................ 58
Hyperlink configuration ............................................................................................... 58
Linked templates ........................................................................................................ 61
Querying WorkItems that are linked to TestResults .................................................... 61
  Using Object Queries to query additional Data for Test Management Objects ........ 62
Console Extension ..................................................................................................... 64
  Installing the Console Extension ............................................................................ 64
  Commands ............................................................................................................... 65
Customization of the Configuration File ...................................................................... 67
FAQ ........................................................................................................................... 71
System Requirements
AIT WordToTFS supports Word 2016 as well as Windows 10. Usually it also works on other combinations, but this configuration is used for as test configuration for WordToTFS.

Make sure you have the following software installed before trying to attempt to install the WordToTFS add-on:

.NET Framework 4.5.2  +  Windows Installer 3.1
+ Visual Studio 2010 Tools for Office Runtime

For more information please go to: https://wordtotfs.aitgmbh.de

Quick Start: Installation
You can choose to directly install WordToTFS via Click Once from our webserver, or to install it through the offline setup. Just follow the installation wizard.

WordToTFS will receive updates automatically as long as you have an active Internet connection.

Just refer to https://wordtotfs.aitgmbh.de for the latest news and updates.

Quick Start: Overview

(1) Connect to a team project in TFS, if connected, shows server and project connected to, click again to disconnect
(2) Publish the changes from the document into the connect team project in TFS
(3) Return on initial state of work item based on state from TFS
(4) Opens the task pane to import existing work items from TFS into the document
(5) Compare work item in the document with work item from TFS
(6) Dropdown to select a mapping template
(7) Ignore formatting
(8) Ignore conflicts
(9) Opens template manager to manage template locations
(10) Inserts a blank work item table
(11) Inserts a work item table with default values
(12) Insert a header into work item
(13) Opens the task pane to edit default values for new work items
(14) Deletes all Ids from published work items to re-publish word file to another server

(15) Opens the Area and Iteration task pane to insert Area and Iteration Paths into the document

(16) Generate test specification report by work item or by query

(17) Generate test result report based on test plan and test suites

(18) **Help** opens help file

(19) **About** shows version of WordToTFS and also shows data related to the AIT company

(20) **Update** starts updating version of WordToTFS

(21) **Feedback** opens a web form to give feedback to AIT

(22) **Settings** opens settings dialog (**Debug settings, Console extension settings**)
Quick Start: Add a Template Location

1. Click the **Template Manager** button

2. Click **Add Source** in the lower
3. Enter name and path of the location
4. Click **Add**

5. Click on the List box entry, check all templates are shown correct

6. Select the template to use for the current document from the Template combo box
Quick Start: Connect, Create, Publish

1. Click **Connect**

2. Choose server

3. Choose Team Project Collection

4. Choose Team Project

5. Click **Connect**

6. Select Template

7. Add some work items

8. Fill in the work item table with values according to your rule set

9. Click **Publish**
Product Usage

Connect to a Team Project

To connect WordToTFS 2010 with a Microsoft Team Foundation Server Team Project, press the **Connect** button in the WordToTFS Ribbon. A new dialog will open as shown in figure 1. Choose a Team Foundation Server, a Collection and a Team Project where the work items will be stored. Press the **Connect** button at the bottom of the dialog to connect the current Word document to the Team Foundation Server.

![Connect to Team Project dialog](image)

Figure 1 - Connect the Word document to a Team Foundation Server

1. Press the **Connect** button in the ribbon.
2. Select a Team Foundation Server from the drop down box or add a new Team Foundation Server by pressing the **Servers...** button.
3. Select a Collection from the list of available Collections in the lower left part of the window.
4. Select a Team Project from the list of available Team Projects in the lower right part of the window.
5. Press **Connect**

Choose a Template

To select an appropriate template just select the right one from the dropdown in the templates section.

Note that the selected template has to match the Team Project type which WordToTFS is connected to. If the selected template contains work items or work item fields that are not supported by the Team Project you will be shown an addition dialog each time you change to this template or connect to another server.

Each template owns a tooltip which tells you about its source location.
After selecting the template, the defined fields in the template will be checked against the server you are connected to. In case of mismatching, Word to TFS will tell you about:

![Missing Mapping Fields Dialog](image)

**Figure 3 - Missing Mapping Fields Dialog**

Even if there are missing fields, you should be able to synchronize with the TFS server. The ability of successfully synchronize depends on the type of missing field. If it is an optional field, the field value just isn’t saved on the TFS (but in the word file, of course) or in the other way round, it’s not resynced to the work item table in Word. However, if it is a required field and the direction is word to TFS, you are not able to successfully sync to the TFS server.

**Manage Available Templates**

You can see the template manager button right next to the template selection dropdown. The template manager provides you independent use of different templates from different locations such as from your network, your local hard drive or even from the internet via http. After opening it, you can see your default template location provided by AIT. Select the list entry, to see further details.

![Template Manager](image)

**Figure 4 - The template manager and the default templates**
To add a new location, click on the add Source expander on the lower side of the panel, enter your template location, and click add. WordToTFS immediately starts to scan the folder for suitable template files, and add them after a quick check to the templates list. As already mentioned the path can be a UNC, local or even a web path. For providing web location ask your administrator.

Now you are able to select the new template from the templates combo box. (Figure 2)

**Insert Work Items**

Depending on the selected Template different Work Item Types are available for insertion into the document.

A new Work Item is inserted as table into the document.

The table contains hints what should be inserted in each cell. The default values can be modified using the *Edit Default Values* task pane.
Support for any kind of work item

WordToTFS 2010 supports any kind of work items. The customizable *.w2t configuration files are capable of describing any kind of work items, those predefined by Microsoft Team Foundation Server as well as custom created work items. In addition a single template may contain any number of different work item types. To select a specific work item type click on the New button to open a selection menu of all available work items for the selected template as shown in figure 3 and figure 4. Selecting one of the available work items will add a new table as defined in the work item configuration into the document and fills the table with default values. The Empty button provides the same functionality like the New button does, except the new created table will not contain any default values.

Work Item Overview

To get to the work item overview press the Compare button. It allows you to compare the work item in your word document to the work item on your server. You can choose to either query all work items found in your word document, or you can pick one of your saved queries. Selecting a query will only compare work items found in both, the query and your document. It will also tell you, if the query returned work items which are not yet imported into your document.

Publish work items to TFS

To publish work items from the Word document to the connected Team Project press the publish button in the WordToTFS ribbon. (This button is only enabled if you are connected to a TFS!) Publishing will create new work items in the Team Project for each work item in the Word document that does not have an id yet. Any work items from the Word document that already have an id will be updated. You can see the state of the publishing process on the right side at a task pane, which opens automatically once you have started the publishing process.

After the publish process has completed, all work items that are successfully saved to TFS are shown as list in the Publish task pane.

In case of any errors during the Publish, a red cross will indicate the error. Click the error entry in the Publish task pane to see details about the issue. To navigate to the work item table that caused the error, use the magnifier icon at the bottom right corner of the error entry.
Figure 12 - Errors during Publish are listed inside the Publish task pane.

Figure 13 - Sample Requirement work item table.

During publish the state, revision and work item id is inserted or updated inside the Word document.

Figure 14 - The published work item in TFS.
Edit default values for the current template

To edit the default values of the current template, press the button **Edit Default Values**. At the right border of the window a new task pane will appear, containing all work items of the selected templates and all their fields.

![Image of Edit Default Values task pane](image)

After editing the default values the window may be closed. Inserting a new work item using the **New** button will create a new table in the word document using the previously specified default values.

All default values can be reset by pressing the **Reset Values** button at the bottom of the default values window.

1. Click the **Edit Default Values** button
2. Edit the default values in the **Default Values** task pane at the right side of the screen
3. Close the **Default Values** task pane

Delete work item ids from an existing document

By pressing the **Delete Ids** button as shown in figure 11, all fields of all tables in the document that points to a TFS work item **System.Id** field are cleared. This allows the synchronization of one document on different Team Foundation Servers.

![Image of Delete Ids button](image)

1. Press the **Delete Ids** button
2. Confirm the deletion of all ids
Import Existing Work Items from TFS

You can import work items from TFS into your Word document by clicking **Get Work Items**. This opens the **Get Work Items** task pane which gives you several options for getting work items from TFS.

You can query work items by Work Item Query, IDs or title. You can also include linked work items based on work item links from the **Hierarchy** pane.

Click **Find** to get the work items matching your criteria. To insert the work items found, select them from the result list and click **Import**. The work item tables will be inserted at the current cursor position within the Word document. In case work item tables already exist in the document, they will be updated (overwritten!) inside the document with the latest values from TFS.

**Create Hyperlink** creates hyperlinks to the selected work items at the cursor position.

After the import, the complete work item table is inserted into the document.
The work item might look like this in TFS:

![Same sample in TFS](image)

**Insert Area and Iteration Path Values**

You can insert Area and Iteration Path values within the document. This way, you will be able to set for instance the planned version or the feature area of your work items from within Word.

Simply click **Area and Iteration Path** from the WordToTFS ribbon tab. This will open the Area and Iteration Path task pane shown on the left.

The Area and Iteration tree from the connected Team Project will be shown. Select an Area or Iteration Path node and click **Insert Area/Iteration** to add it at the current text cursor position into the document. This could be inside a work item table:

![The Area and Iteration task pane](image)
Refresh all or selected work items

In order to update all the work items inside the document with the latest contents from TFS, use the Refresh option from the WordToTFS ribbon tab.

You can also refresh individual work items by selecting the tables in Word and execute “Refresh selected work items.” from the drop-down Refresh menu.

Using Headers

Headers are document elements like work items that can be inserted anywhere outside of a table. Headers are used to set field values for multiple work items at once. Whether a work item has its field values set by a header depends on the position of the work item. Only work items in the header range are affected by its configuration and values.

The range of a header starts with the header itself and ends with a header of the same or a lower hierarchy level. Headers can be used to set fields for all work items within its range. This is especially useful if the fields of the header are not also mapped in the affected work items as it helps keeping the document more compact. You can for example define a header that only sets the area path for all work items and don’t have to add the area path field to other work item mappings. The button to insert headers is only visible if there are headers configured in the selected template.

Example: With a correctly configured header, publishing the change request will automatically set the Area defined in the header for this work item. This helps keeping the document clean and easy to maintain, and is especially helpful when adding many items.

Provide Feedback
Contact us!

Test Management Reports

WordToTFS supports the generation of test specification reports and test result reports based on the information stored in TFS. The features are accessible via the buttons in the Test Management Ribbon group. Use the buttons shown in Figure 29 to show or hide one of the Test Management task panes.

Test specification report

The test specification reports enables the creation of reports that lists test cases and related information. Once the information is received from TFS and inserted into word it is disconnected from TFS again. Also the refresh option is not available for generated reports. The task pane allows to configure how the report is generated. The following table explains the different options:

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Test Plan</td>
<td>Allows the user to select the test plan that contains the test cases that should be used in the report.</td>
</tr>
<tr>
<td>Select Test Suite</td>
<td>The test suite hierarchy of the selected test plan is shown and the user has the option to restrict the test cases to use in the report.</td>
</tr>
<tr>
<td>Create document structure</td>
<td>This option enables the creation of word document structure.</td>
</tr>
<tr>
<td>Document structure</td>
<td>If the report should be structured, it is possible to choose one of these possibilities:</td>
</tr>
<tr>
<td></td>
<td>• Iteration Path</td>
</tr>
<tr>
<td></td>
<td>• Area Path</td>
</tr>
<tr>
<td></td>
<td>• Test Plan Hierarchy</td>
</tr>
<tr>
<td>Skip Levels</td>
<td>If the report should be structured and the structure is either Iteration Path or Area Path, this number determines how many levels should be skipped in generated structure. See also chapter Select Builds – Filtering based on build association</td>
</tr>
<tr>
<td></td>
<td>This option is only available for XAML-based builds. The new build Engine concept introduced with TFS 2015 is not supported. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build quality assigned, which is configured in the WordToTFS-template. Please check the WordToTFS-template “CMMI(2015)” for details.</td>
</tr>
</tbody>
</table>

Figure 29: Test reporting ribbon

Test Management Reports

WordToTFS supports the generation of test specification reports and test result reports based on the information stored in TFS. The features are accessible via the buttons in the Test Management Ribbon group. Use the buttons shown in Figure 29 to show or hide one of the Test Management task panes.

Test specification report

The test specification reports enables the creation of reports that lists test cases and related information. Once the information is received from TFS and inserted into word it is disconnected from TFS again. Also the refresh option is not available for generated reports. The task pane allows to configure how the report is generated. The following table explains the different options:

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Test Plan</td>
<td>Allows the user to select the test plan that contains the test cases that should be used in the report.</td>
</tr>
<tr>
<td>Select Test Suite</td>
<td>The test suite hierarchy of the selected test plan is shown and the user has the option to restrict the test cases to use in the report.</td>
</tr>
<tr>
<td>Create document structure</td>
<td>This option enables the creation of word document structure.</td>
</tr>
<tr>
<td>Document structure</td>
<td>If the report should be structured, it is possible to choose one of these possibilities:</td>
</tr>
<tr>
<td></td>
<td>• Iteration Path</td>
</tr>
<tr>
<td></td>
<td>• Area Path</td>
</tr>
<tr>
<td></td>
<td>• Test Plan Hierarchy</td>
</tr>
<tr>
<td>Skip Levels</td>
<td>If the report should be structured and the structure is either Iteration Path or Area Path, this number determines how many levels should be skipped in generated structure. See also chapter Select Builds – Filtering based on build association</td>
</tr>
<tr>
<td></td>
<td>This option is only available for XAML-based builds. The new build Engine concept introduced with TFS 2015 is not supported. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build quality assigned, which is configured in the WordToTFS-template. Please check the WordToTFS-template “CMMI(2015)” for details.</td>
</tr>
</tbody>
</table>

Figure 29: Test reporting ribbon

Test Management Reports

WordToTFS supports the generation of test specification reports and test result reports based on the information stored in TFS. The features are accessible via the buttons in the Test Management Ribbon group. Use the buttons shown in Figure 29 to show or hide one of the Test Management task panes.

Test specification report

The test specification reports enables the creation of reports that lists test cases and related information. Once the information is received from TFS and inserted into word it is disconnected from TFS again. Also the refresh option is not available for generated reports. The task pane allows to configure how the report is generated. The following table explains the different options:

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Test Plan</td>
<td>Allows the user to select the test plan that contains the test cases that should be used in the report.</td>
</tr>
<tr>
<td>Select Test Suite</td>
<td>The test suite hierarchy of the selected test plan is shown and the user has the option to restrict the test cases to use in the report.</td>
</tr>
<tr>
<td>Create document structure</td>
<td>This option enables the creation of word document structure.</td>
</tr>
<tr>
<td>Document structure</td>
<td>If the report should be structured, it is possible to choose one of these possibilities:</td>
</tr>
<tr>
<td></td>
<td>• Iteration Path</td>
</tr>
<tr>
<td></td>
<td>• Area Path</td>
</tr>
<tr>
<td></td>
<td>• Test Plan Hierarchy</td>
</tr>
<tr>
<td>Skip Levels</td>
<td>If the report should be structured and the structure is either Iteration Path or Area Path, this number determines how many levels should be skipped in generated structure. See also chapter Select Builds – Filtering based on build association</td>
</tr>
<tr>
<td></td>
<td>This option is only available for XAML-based builds. The new build Engine concept introduced with TFS 2015 is not supported. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build quality assigned, which is configured in the WordToTFS-template. Please check the WordToTFS-template “CMMI(2015)” for details.</td>
</tr>
</tbody>
</table>

Figure 29: Test reporting ribbon
Select Builds – Filtering based on build-filters
This option is available for all builds. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build filter assigned, which is configured in the WordToTFS-template. If <BuildFilters> exist in template, then separated <BuildQualities> is not allowed. <BuildAge> represent the age of build relative to today, e.g. if value is 21, only builds younger than 21 days will be shown. Filters <BuildName> and <BuildAge> are available for all builds, until <BuildQualities> is only available for XAML-based builds and <BuildTags> is only available for vNext builds.

Structure of generated report.

<table>
<thead>
<tr>
<th>Include Test Configurations</th>
<th>This option defines whether the test configurations should be included in the report. The corresponding selection allows to select the position of the test configuration, with one of the following possibilities:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort Test Cases by</td>
<td>Defines the criteria the test cases are sorted when added to the report. Available sort types are:</td>
</tr>
<tr>
<td></td>
<td>• Iteration Path</td>
</tr>
<tr>
<td></td>
<td>• Area Path</td>
</tr>
</tbody>
</table>
The test specification reports can also be produced by queries. The task pane shown in Figure 30 is available as sub function of the test specification report button. It looks similar to the “Get Work Items” task pane in Figure 20 and offers the same possibilities to load or define queries. In contrast to the simple test specification report, here only work items can be used in the report (thus no test plan or test suite overview). Thereby it is important to know that only Test Cases and Shared Steps can be included in the report. Further both (Test Cases and Shared Steps) may be used together in a query but report creation can only be done when a single type is used.

However Test Cases are displayed as defined in the Test specification report configuration, whereas Shared Steps are displayed as defined in the field mappings (see section Defining field mappings). Clicking the button “Create Report” will start the generation of the report. The generation is started at the current cursor position. The report includes all checked work items. The order is defined by the query. Once the information is received from TFS and inserted into word, it is disconnected from TFS which means any changes you do to the generated word content cannot be published to TFS again. Also the refresh option is not available for generated reports.

Test result report
The test result report enables the creation of reports that lists the result of different test runs. The following table lists the options available in the test result report task pane:
Term | Explanation
--- | ---
Select Build | Allows to set a filter for the plans and test suites available to select. The available builds are configurable. For details please check the following sub chapter (Select Builds – Filtering based on build association).
Select Test Plan | Allows the user to select the test plan that contains the test results that should be used in the report.
Select Test Suite | The test suite hierarchy of the selected test plan is shown and the user has the option to restrict the test cases to use in the report.
Select Test Configuration | Allows to restrict the test results used in the report to those being generated with a specific test configuration.
Create document structure | This option enables the creation of word document structure.

Document structure | If the report should be structured, it is possible to choose one of these possibilities:
- Iteration Path
- Area Path
- Test Plan Hierarchy

Skip Levels | If the report should be structured and the structure is either Iteration Path or Area Path, this number determines how many levels should be skipped in generated structure. See also chapter Select Builds – Filtering based on build association
This option is only available for XAML-based builds. The new build Engine concept introduced with TFS 2015 is not supported. In this selection field,
not all builds will be displayed, only those which are within the current connected project and have the certain build quality assigned, which is configured in the WordToTFS-template. Please check the WordToTFS-template “CMMI(2015)” for details.

```xml
<MappingConfiguration>
  <TestConfiguration>
    <TestSpecificationConfiguration ... />
    <TestResultConfiguration>
      <BuildQualities>
        <BuildQuality>Released</BuildQuality>
        <BuildQuality>Passed</BuildQuality>
        <BuildQuality>Ready</BuildQuality>
      </BuildQualities>
    </TestResultConfiguration>
  </TestConfiguration>
</MappingConfiguration>

Select Builds – Filtering based on build-filters
This option is available for all builds. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build filter assigned, which is configured in the WordToTFS-template. If <BuildFilters> exist in template, then separated <BuildQualities> is not allowed. <BuildAge> represent the age of build relative to today, e.g. if value is 21, only builds younger than 21 days will be shown. Filters <BuildName> and <BuildAge> are available for all builds, until <BuildQualities> is only available for XAML-based builds and <BuildTags> is only available for vNext builds.

```xml
<MappingConfiguration>
  <TestConfiguration>
    <TestSpecificationConfiguration ... />
    <TestResultConfiguration>
      <BuildFilters>
        <BuildAge>21</BuildAge>
        <BuildNames>
          <BuildName>Name of XAML build</BuildName>
          <BuildName>Name of vNext build</BuildName>
        </BuildNames>
        <BuildTags>
          <BuildTag>Ready</BuildTag>
          <BuildTag>Test</BuildTag>
        </BuildTags>
        <BuildQualities>
          <BuildQuality>Released</BuildQuality>
          <BuildQuality>Passed</BuildQuality>
          <BuildQuality>Ready</BuildQuality>
        </BuildQualities>
      </BuildFilters>
      </TestResultConfiguration>
  </TestConfiguration>
</MappingConfiguration>

Structure of generated report.

<table>
<thead>
<tr>
<th>Include Test Configurations</th>
<th>This option defines whether the test configurations should be included in the report. The corresponding selection allows to select the position of the test configuration, with one of the following possibilities:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AboveTestPlan</td>
<td></td>
</tr>
<tr>
<td>BeneathTestPlan</td>
<td></td>
</tr>
</tbody>
</table>
- BeneathTestSuites
- BeneathFirstTestSuite

Sort Test Cases by

Defines the criteria the test cases are sorted when added to the report. Available sort types are:
- Iteration
- Area Path
- Work Item Id

Include only the most recent test result

If checked, for each test case only the most recent test result (DateCompleted) is reported.

For all selected configurations

If checked, the most recent test result is included for the selected test configurations.

Create Report

Clicking this button will start the generation of the report. The generation is started at the current cursor position.

Select Builds – Filtering based on build association

This option is only available for XAML-based builds. The new build Engine concept introduced with TFS 2015 is not supported. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build quality assigned, which is configured in the WordToTFS-template. Please check the WordToTFS-template “CMMI(2015)” for details.

```xml
<MappingConfiguration>
  <TestConfiguration>
    <TestSpecificationConfiguration ... />
    <TestResultConfiguration>
      <BuildQualities>
        <BuildQuality>Released</BuildQuality>
        <BuildQuality>Passed</BuildQuality>
        <BuildQuality>Ready</BuildQuality>
      </BuildQualities>
    </TestResultConfiguration>
  </TestConfiguration>
</MappingConfiguration>
```

Select Builds – Filtering based on build-filters

This option is available for all builds. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build filter assigned, which is configured in the WordToTFS-template. If <BuildFilters> exist in template, then separated <BuildQualities> is not allowed. <BuildAge> represent the age of build relative to today, e.g. if value is 21, only builds younger than 21 days will be shown. Filters <BuildName> and <BuildAge> are available for all builds, until <BuildQualities> is only available for XAML-based builds and <BuildTags> is only available for vNext builds.

```xml
<MappingConfiguration>
  <TestConfiguration>
    <TestSpecificationConfiguration ... />
    <TestResultConfiguration>
      <BuildFilters>
        <BuildAge>21</BuildAge>
        <BuildNames>
          <BuildName>Name of XAML build</BuildName>
          <BuildName>Name of vNext build</BuildName>
        </BuildNames>
      </BuildFilters>
    </TestResultConfiguration>
  </TestConfiguration>
</MappingConfiguration>
```
Structure of generated report

The following section details how the word document structure is generated based on the users selection in the test reporting task panes.

Let’s suppose the test cases are defined in following iterations:

- **Base** – 2 test cases (BaseTC1 and TaseTC2)
- **Base/Extended** – 0 test cases
- **Base/Extended/Iteration 1** – 3 test cases (Iteration1TC1, Iteration1TC2 and Iteration1TC3)
- **Base/Extended/Iteration 2** – 1 test case (Iteration2TC1)
- **Base/Extended/Iteration 3** – 0 test cases

If the Skip Levels is set to 0, the structure of the generated report will be as shown in Figure 32.

**Notes**

- In generated report are used all test cases defined in selected test suite and all its child test suites.
- If the iteration contains no test case, header for test case block is not generated.
- If the iteration contains no test case and its child iteration too, iteration is not used in generated report.
If the Skip Levels is set to 2, the structure of the generated report will be as shown in Figure 33.

**Notes**
- If the block is not shown, its test cases are not shown too.

A structure based on the area path could be generated in a way equal to a structure based on iteration path.

Let’s suppose the test cases are defined in following iterations:

- **Backend** – 0 test cases
- **Backend/Database** – 2 test cases (DatabaseTC1 and DatabaseTC2)
- **Backend/Services** – 3 test cases (ServicesTC1, ServicesTC2 and ServicesTC3)

If the Skip Levels is set to 0, the structure of the generated report will be as follows:

![Diagram of test specification report structured by area and skip levels is set to 0]

Figure 34: Test specification report structured by area and skip levels is set to 0

If the Skip Levels is set to 2, the structure of the generated report will be as follows:

![Diagram of test specification report structured by area and skip levels is set to 2]

Figure 35: Test specification report structured by area and skip levels is set to 2
Note
- If no area exists for the filter, no test case is generated.

The last criteria for document generating a document structure is the test plan hierarchy. Supposed the test suites are defined in followed structure we would get a document structure as shown in Figure 36.

- **Common** – 1 test case (CommonTC1)
- **Publish** – 2 test cases (PublishTC1 and DatabaseTC2)
- **Internet** – 1 test case (InternetTC1)
- **Intranet** – 1 test case (IntranetTC1)
- **Conflicts** – 2 test cases (ConflictsTC1)

A final scenario is the unstructured report where no document structure is generated

For the test result report the same mechanism is used but there are some additional restrictions:

1. The test configurations being used are inserted on the beginning of report.
2. Only test cases with at least one test result are included in the report.
3. Under each test case are displayed its test results.
Enhanced Document Structure

The functionalities of WordToTFS can also be used to visualize any parent-child relationship of work items in Team Foundation Server. Therefore custom templates can be assigned to each level of a work item hierarchy, each with its formatting. During a query, WordToTFS is able to determine the level of work items, depending on their relationship between each other. To achieve a custom formatting several steps have to be performed.

- The name of the relationship (link type) that should be analyzed and visualized has to be specified in the `MappingConfiguration` of the w2t file using the property `TypeOfHierarchyRelationships`.
- A custom Mapping has to be created for each level. The level has to be specified in the property `WorkItemSubTypeField HierarchyLevel` and its corresponding value of the `WorkItemSubTypeValue`. For example the topmost item has the HierarchyLevel “0” and its succeeding child has therefore the HierarchyLevel “1”.
- Each Mapping can contain a customized template file with its own formatting. (See Chapter: [Customizing the template layout: .xml files](#))

During a query execution, WordToTFS builds an internal hierarchy and applies the customized templates to the work items.

**Example Configuration to visualize a relationship**

The configuration to visualize the relationship of `Parents` would look like this:

```
<MappingConfiguration
  ...
  TypeOfHierarchyRelationships = "Parent"
  ...
  ... />
```

The corresponding configuration of the Mapping items would look like this:

```
<Mapping Name="Feature"
  WorkItemSubTypeField="HierarchyLevel"
  WorkItemSubtypeValue="0"
  RelatedTemplate="MSFForCMMI(2010).Feature_Level0.xml"
  HideElementInWord="true"
  AssignRegularExpression="Feature_Level0"
  ...
  ... />
```

```
<Mapping Name="Feature"
  WorkItemSubTypeField="HierarchyLevel"
  WorkItemSubtypeValue="1"
  RelatedTemplate="MSFForCMMI(2010).Feature_Level1.xml"
  HideElementInWord="true"
  AssignRegularExpression="Feature_Level1"
  ...
  ... />
```

The corresponding layout could look like the following:
The formatting of the title will be applied, when the work item is imported from TFS.

**Important Notes & Limitations**

- The determination of the HierarchyLevel works only for equal work items. If a child work item has a different type than its parent, it has also the HierarchyLevel 0.
- WordToTFS determines the HierarchyLevel for the work items of the current query. The items are not automatically sorted, an appropriate sorting must be specified in the query itself.
- If the table structure of the different levels should differ from each other, special attention has to be paid that the **AssignRegularExpression** of the template is different for each mapping work item.
- If relationships on TFS are changed a new get of all items is necessary. Currently the refresh function of WordToTFS cannot rearrange the items.

**Hint:** You can use the new property **HideElementInWord** in your Mapping to hide a Mapping item from the New Item Selection, this will help to keep your WordToTFS menu clean.

**Managing Templates**

WordToTFS comes with some default templates, which are shown under the “AIT GmbH” location in the templates manager. The template manager allows you to easily work with different configurations from different physical locations. The templates found at the given locations are cached, so you can use them even if the original source is not available.

**Add a Template Location**

When you add a template, you need to provide a directory as well as a name used to identify this template in the template manager and the template dropdown in the Word ribbon.

**Local and UNC location**

A typical location folder contains 3 different types of files. (See later in this document how to customize them) First of them the “w2t” file, one per template. The w2t file contains the mapping between the TFS server fields and the Word table cells. We deliver WordToTFS with the following templates per default: Microsoft Visual Studio Scrum, MSF for CMMI, MSF for Agile and TeamSystemPro(2010). The second type is the word-table files as .xml. They contain the design and look of each of the work item types, such as task, requirement, bug etc. The last one is a location avatar as ico file. If you want to show an icon in the template manager you have to add it to the location folder and rename it to “favicon.ico”.

Add a location at the template manager:
Web locations

[Unfortunately, this isn’t supported anymore – we’re looking for customers that will fund this feature. Please contact: AIT.vertrieb@aitgmbh.de in case you want to finance its development.]

If you want to provide a worldwide available web location, you need a little bit of extra effort. The location folder is just the same as shown above, but you have to link to an additional mapping file as shown below. Link to these files full web path, like this:

![Figure 41 - An example of a configuration xml file](image)

The referenced xml file contains a list of w2t files. Like the local or UNC location, these w2t files are the representation of the specific templates. The web xml file may look like this:

```xml
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfTemplatesConfigInfo xmlns="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsi="http://www.w3.org/2001/XMLSchema">
  <WebTemplatesConfigInfo Path="HTTP://www.aitgmbh.de/templates/AITTempCMMI(2010).w2t"/>
  <WebTemplatesConfigInfo Path="HTTP://www.aitgmbh.de/templates/TeamSystemPro(2010).w2t"/>
</ArrayOfTemplatesConfigInfo>
```

Team Project specific template

[This feature has not yet been implemented – it’s a product idea only. We’re looking for customers that will fund this feature. Please contact: AIT.vertrieb@aitgmbh.de in case you want to finance its development.]

Team project specific templates are set by checking in the template files into TFS source control as a project sub folder e.g. $/MyTeamProject/WordToTFSTemplates/myTemplate.w2t.

When a user connects with the Team Project via WordToTFS, the template checked into source control will be shown as available templates in WordToTFS only. All other templates will be hidden. This helps users using the right template and prevents them from having to configure a custom template location and choosing the template from the drop-down.

Project mapped template source.
Adding a location with subfolders will give you the option to set up the location as project mapped template source. This feature allows preselecting available templates based on what project the user is connected to. The expected folder structure relative to the location is “server name\team project collection name\project name”. A template placed in the subfolder “Server\DefaultCollection\Project A” will only be available if the user is connected to the project “Project A” within the Team Project collection “DefaultCollection” on the server “Server”.

Customizing the template configuration: w2t file

[In case you need further information on template customization or want to have a custom template implemented by us please contact: AIT.vertriebe@aitgmbh.de]

WordToTFS templates are fully customizable. Template configurations are XML files saved in files with a “w2t” extension. A template configuration may contain button customization configurations, work item mapping configurations, header configurations and test report configurations. A new configuration is defined using the MappingConfiguration XML element. A configuration defines behavior that applies to each work item and also allows enabling and disabling UI elements when this configuration is selected.

Table 1: Attributes of the MappingConfiguration element

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoConnect(true/false)</td>
<td>Determines if WordToTFS should try to connect automatically upon startup. This works only if the template is set as default (See DefaultMapping)</td>
</tr>
<tr>
<td>AutoRefreshQuery</td>
<td>A query that will be refreshed immediately after connection to TFS is established. The resulting work items will be automatically imported into the document. Existing work items in the document will be replaced. Please state the query including the whole path. <strong>Example:</strong> AutoRefreshQuery= [ProjectName]/Shared Queries/[NameOfYourQuery] If you use this attribute in combination with AutoConnect, a query will be refreshed every time Word starts.</td>
</tr>
<tr>
<td>DefaultMapping (true/false)</td>
<td>Determine if the template should be selected as default template. If multiple templates have the DefaultMapping set to true, the first template is choosen as default.</td>
</tr>
<tr>
<td>DefaultServerUrl</td>
<td>Default server that WordToTFS should use during connect. If this property is set, the server and project cannot be selected manually anymore. Sample url: http(s)://[YourServer]:[Port]/tfs/[YourCollection]/</td>
</tr>
<tr>
<td>DefaultProjectName</td>
<td>Default project that WordToTFS should use during connect. Use in combination with DefaultServerUrl.</td>
</tr>
<tr>
<td>ShowName</td>
<td>Name of the template presented to the user.</td>
</tr>
<tr>
<td>UseStackRank (true</td>
<td>false)</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IgnoreFormatting (true</td>
<td>false)</td>
</tr>
<tr>
<td>ConflictOverwrite (true</td>
<td>false)</td>
</tr>
<tr>
<td>EnableRefresh, EnablePublish, EnableGetWorkItems, EnableOverview, EnableHistoryComment, EnableEmpty, EnableNew, EnableEditDefaultValues, EnableDeleteIds, EnableArealerationPath, EnableIgnoreFormattingSwitch, EnableTemplateManager, EnableTemplateSelection, EnableConflictOverwriteSwitch (true</td>
<td>false)</td>
</tr>
<tr>
<td>TypeOfHierarchyRelationships</td>
<td>Specifies the name of any relationship between Work Items that is analyzed by word to TFS. (See Enhanced Document Structure)</td>
</tr>
<tr>
<td>PreOperations, PostOperations</td>
<td>The definition of an array of operation that should be performed prior to insertion of the template / after inserting the template to target document. The Operation is performed in target document.</td>
</tr>
<tr>
<td>Operation</td>
<td>Type of the operation defines the operation to execute. Possible types:&lt;br&gt; - InsertParagraph&lt;br&gt; - MoveCursortToStart&lt;br&gt; - MoveCursortToEnd&lt;br&gt; - DeleteCharacterLeft&lt;br&gt; - DeleteCharacterRight&lt;br&gt; - MoveCursorToLeft</td>
</tr>
</tbody>
</table>
GetDirectLinksOnly

Default: false

Determines whether to get direct links only when executing Get Work Items By ID with the option enabled to Include linked Work Items.

When set to false, the complete chain of links is traversed until there is no further link that matches the specified link type. Therefore a great amount of work items has to be traversed especially when there is no filter on the link type. This can lead to long loading times.

The following example shows the result of setting the parameter value either to false or true when entering work item ID 21.

Scenario:

- When set to false, all work items (21, 30, 33, 34, 36 and 37) are returned.
- When set to true, only the work items 21, 30, 33 and 37 are returned.

Example:

```xml
<MappingConfiguration ShowName="TeamSystemPro(2010)"
    RelatedSchema="MSFForCMMI.Schema.xsd"
    DefaultMapping="false"
    EnableRefresh="true"
    UseStackRank="false"
    IgnoreFormatting="true"
    GetDirectLinksOnly="true">
    <PreOperations>
        <Operation Type="InsertNewPage"/>
    </PreOperations>
    <PostOperations>
        <Operation Type="RefreshAllFieldsInDocument"/>
    </PostOperations>
</MappingConfiguration>
```

Customizing the buttons of the ribbon

It is possible to customize the text shown on the buttons of WordToTFS. Normally the pane shown in Fig. 1 is displayed, whereas the highlighted buttons can be customized.
Table 2: Attributes of the Button XML element

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| Name      | Name of the button, which should be customized. The available buttons are:  
- Publish  
- Refresh  
- Get Work Items  
- Compare  
- Empty Work Item  
- New Work Item  
- Insert Header  
- Default Values  
- Delete Ids  
- Area und Iteration Path  
- Test Spec Report  
- Test Spec Report By Query  
- Test Result Report |
| Text      | Text, which should be displayed on the selected button. |

Example:

```xml
<ButtonsCustomization>
  <Button Name="Publish" Text="Publish Text" />
  <Button Name="Refresh" Text="Refresh Text" />
</ButtonsCustomization>
```

Creating a new work item mapping

A work item mapping relates to a single work item type (Requirement, Task, etc.) and defines what fields of that work item are mapped on what cells of a given work item table. It also defines how those fields are synchronized when the user refreshes or publishes a document. A work item mapping is created using the Mapping XML element within a MappingConfiguration XML element.

Table 3: Attributes of the Mapping XML element

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkItemType (mandatory)</td>
<td>The TFS work item type (“Requirement”, “Bug”, etc.) this mapping applies to.</td>
</tr>
<tr>
<td>MappingWorkItemType</td>
<td>The name of the work item type in case you want to use a different name in your word document.</td>
</tr>
<tr>
<td>RelatedTemplate (mandatory)</td>
<td>The xml table style file. See below on how to customize these.</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>AssignRegularExpression (mandatory)</td>
<td>A regular expression to find the MappingWorkItemType. The text you type in here must match text in a table cell in order for WordToTFS being able to identify the table as particular work item type.</td>
</tr>
<tr>
<td>AssignCellRow (mandatory)</td>
<td>The row of the cell where AssignRegularExpression is checked. The cell should contain the AssignRegularExpression as text in the template. Row count starts at 1.</td>
</tr>
<tr>
<td>AssignCellCol (mandatory)</td>
<td>The column of the cell where AssignRegularExpression is checked. The cell should contain the AssignRegularExpression as text in the template. Column count starts at 1.</td>
</tr>
<tr>
<td>ImageFile (mandatory)</td>
<td>An image to show in “new” button. If not customized, just type “standard.png”</td>
</tr>
<tr>
<td>WorkItemSubtypeField</td>
<td>This property offers the functionality to specify subtypes of the current Mapping.</td>
</tr>
<tr>
<td>WorkItemSubtypeValue</td>
<td>The corresponding value for the WorkItemSubtypeField.</td>
</tr>
<tr>
<td>HideElementInWord</td>
<td>Optional: Can be true or false and determines if this Mapping should be visible in the New and Empty Work Item buttons. If not specified, the default value is false (See: Chapter Support for any kind of work items).</td>
</tr>
<tr>
<td>PreOperations, PostOperations</td>
<td>The definition of an array of operation that should be performed prior to insertion of the work item / after inserting the item to target document. The Operation is performed in target document.</td>
</tr>
<tr>
<td>Operation</td>
<td>Type of the operation defines the operation to execute. Possible types:</td>
</tr>
<tr>
<td></td>
<td>• InsertParagraph</td>
</tr>
<tr>
<td></td>
<td>• MoveCursorToStart</td>
</tr>
<tr>
<td></td>
<td>• MoveCursorToEnd</td>
</tr>
<tr>
<td></td>
<td>• DeleteCharacterLeft</td>
</tr>
<tr>
<td></td>
<td>• DeleteCharacterRight</td>
</tr>
<tr>
<td></td>
<td>• MoveCursorToLeft</td>
</tr>
<tr>
<td></td>
<td>• MoveCursorToRight</td>
</tr>
<tr>
<td></td>
<td>• InsertNewPage</td>
</tr>
<tr>
<td></td>
<td>• RefreshAllFieldsInDocument</td>
</tr>
</tbody>
</table>

Example:

```xml
<MAPPING WorkItemType="Task">
  <MappingWorkItemType="Task">
    <RelatedTemplate>TeamSystemPro(2010).Task.xml</RelatedTemplate>
    <AssignRegularExpression>Task</AssignRegularExpression>
    <AssignCellRow>1</AssignCellRow>
    <AssignCellCol>1</AssignCellCol>
    <ImageFile>standard.png</ImageFile>
  </MappingWorkItemType>
  <PreOperations>
    <Operation Type="InsertNewPage"/>
  </PreOperations>
</MAPPING>
```
**Defining field mappings**

A field mapping defines what work item field is mapped onto which cell of a template and when to refresh or publish the value of this cell. Field mappings are created using the *Field* XML element within the *Mapping* XML element. The fields *System.ID* and *System.Rev* must be mapped in order for WordToTFS to work correctly!

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><em>ReferenceName</em> of the field in your process template, such as ‘System.Id’</td>
</tr>
<tr>
<td><strong>FieldValueType</strong></td>
<td>The type of this field. This setting mostly depends on the data type of this field in your process template. For details see below.</td>
</tr>
<tr>
<td><strong>Direction</strong></td>
<td>The direction in which to synchronize this field. For possible values and their meaning, see table below.</td>
</tr>
<tr>
<td><strong>MappingTableRow</strong></td>
<td>The row of the word table cell this field is mapped to. Row count starts at 1.</td>
</tr>
<tr>
<td><strong>MappingTableCol</strong></td>
<td>The column of the word table cell this field is mapped to. Column count starts at 1.</td>
</tr>
<tr>
<td><strong>WordBookmark</strong></td>
<td>Inserts a bookmark in the specified table cell.</td>
</tr>
<tr>
<td><strong>TestCaseStepDelimiter</strong></td>
<td>A string that separates test step title and test step expected result when mapping test steps of test case work items. Default value is ‘-‘.</td>
</tr>
<tr>
<td><strong>ShapeOnlyWorkaroundMode</strong></td>
<td>This attribute can only be set for HTML field, because HTML field can contains inline shape (picture). If <em>ShowAsError</em> is set, then word will show an error, in case when trying to publish a field with only an inline shape (picture) and no text. Field without text word will not export shapes. However, if <em>AddSpace</em> is set word will automatically add a space after the inline shape instead of showing an error. Default value is <em>AddSpace</em>.</td>
</tr>
</tbody>
</table>
| **DateTimeFormat**            | The date time will be formatted by the assigned value. If the field is a date time and this attribute is not used, the standard date time format will be used including hours, minutes and seconds. Use the following values for formatting: „MM/dd/yy H:mm:ss“.

**Example:**

This configuration maps the id of a work item to the first column in the second row of a table and inserts a bookmark. The id is only read from the server, but never written to.

```xml
<Field Name="System.Id"
   FieldValueType="PlainText"
   Direction="TfsToOther"
   MappingTableRow="2"
   MappingTableCol="1"
   WordBookmark="BookmarkName"
/>`
**Direction Attribute**

The exact behavior of when field values are published or refreshed is given by the following table:

<table>
<thead>
<tr>
<th></th>
<th>Get</th>
<th>Refresh</th>
<th>Publish</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetOnly</strong></td>
<td>Value is refreshed</td>
<td>-</td>
<td>Value is refreshed once for work items that have been created in Word.</td>
</tr>
<tr>
<td><strong>TfsToOther</strong></td>
<td>Value is refreshed</td>
<td>Value is refreshed</td>
<td>Value is refreshed</td>
</tr>
<tr>
<td><strong>OtherToTfs</strong></td>
<td>Value is refreshed</td>
<td>Value is refreshed</td>
<td>Value is <strong>published</strong>, then refreshed</td>
</tr>
<tr>
<td><strong>PublishOnly</strong></td>
<td>-</td>
<td>-</td>
<td>Value is <strong>published</strong></td>
</tr>
<tr>
<td><strong>SetInNewTfsWorkItem</strong></td>
<td>Value is refreshed</td>
<td>Value is refreshed</td>
<td>Value is <strong>published</strong> once for work items that have been created in word. Value is refreshed.</td>
</tr>
</tbody>
</table>

**FieldValueType Attribute**

**PlainText**

The field value is interpreted as plain text both in Word and in the Team Foundation Server. Use this value for all fields that are configured as simple data types like Integer, String, etc. in your process template.

**HTML**

The field value is interpreted as HTML both in Word and in the Team Foundation Server. Use this value for all fields that are configured as HTML data type in your process template. If you do not need the HTML formatting options and have encountered problems you might want to use PlainText instead.

**DropDownList**

The field value is interpreted as Dropdown-List in Word. Use this value for all fields that are configured with AllowedValues, either explicitly or implicitly (e.g. ‘System.AssignedTo’) in your process template.

For non-required fields there is also a blank line to choose in the list of entries, which deletes the current selected value.

**BasedOnVariable**

The field value is interpreted as a variable, which is defined inside the MappingConfigurations. Such a variable can be used to insert additional text into tablefields.

```xml
<Field Name="VersionInfo"
   FieldValueType="BasedOnVariable"
   VariableName="Replace"
   MappingTableRow="3"
   MappingTableCol="1"/>
```

**Note:** This field type is excluded from synchronization. Replacement is only done on get or refresh work items.

**BasedOnSystemVariable**
Currently only one system variable, namely \textit{WordToTFS.Version}, is defined, which provides the WordToTFS-version currently in use. The variable can be used to insert this additional information into table fields.

\begin{verbatim}
<Field Name="VersionInfo"
   FieldValueType="BasedOnSystemVariable"
   VariableName="WordToTFS.Version"
   MappingTableRow="3"
   MappingTableCol="1"/>
\end{verbatim}

\textit{BasedOnFieldType}

The field value is interpreted as numbered list of test case titles and test case expected results, separated by the \textit{TestCaseStepDelimiter}. Use this value for all fields that are configured as \textit{TestStepsControl} data type in your process template (e.g. ‘Microsoft.VSTS.TCM.Steps’). Please note that formatting of test steps is not supported in WordToTFS and will be lost during import or publish!

\textbf{Example configuration:}

\begin{verbatim}
<Field Name="Microsoft.VSTS.TCM.Steps"
   FieldValueType="BasedOnFieldType"
   Direction="OtherToTfs"
   MappingTableRow="3"
   MappingTableCol="1"
   TestCaseStepDelimiter="->" />
\end{verbatim}

\begin{table}
\begin{tabular}{|l|l|l|}
\hline
Test Case & 2 & Sven Hubert & Design \tabularnewline
\hline
First Test & & 311 & 3 \tabularnewline
\hline
1. Do that & & & \tabularnewline
2. Do this -> Check that & & & \tabularnewline
3. Do more & & & \tabularnewline
4. Close all & & & \tabularnewline
\hline
\end{tabular}
\end{table}

Figure 42 – Work item with mapped test case test steps in Word
Figure 43 - Work item with test steps in MTM

**Default value**
If you want to provide a default value to a cell/field, you have to add the sub node “DefaultValue”.
On creating a new work item at word, these values are used to prefill the table, and you also be able
to edit the default values through the “Edit default Values” button.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Name</td>
<td>The label used in default values pane</td>
</tr>
<tr>
<td>&gt;Value&lt;</td>
<td>The default value to use</td>
</tr>
</tbody>
</table>

Example:
```
<DefaultValue ShowName="Title">Task Title</DefaultValue>
```

**OLE Support**
By default, OLE objects are converted into images which are attached to the work item. This behavior
can be changed by setting the attribute ‘HandleAsDocument’ to true. This causes the complete field
contents to be attached as micro Word document to the work item. The work item field will contain
a view in HTML which should not be changed by the user in TFS as it will be overwritten during the
next publish in Word. The micro Word document contains all original OLE objects and is used during
import and refresh in Word.

The attribute ‘HandleAsDocumentMode’ can be set to ‘OleOnDemand’ which reduces the behavior
of attaching a micro Word document to the cases where OLE objects are actually contained in the
work item table.

Example:
```
<Field Name="System.Description"
    FieldValueType="HTML"
    Direction="OtherToTfs"
    MappingTableRow="3"
```
Example for OLE Support using an equation
From a technical point of view equations in Word are OLE objects. To get a formula into a work item, just add an equation within the description field. Reminder: It will be editable only from Word.

After publishing the work item to TFS, the formula will be shown as a picture in the TFS web view. The whole content of the description including the formula is separately stored within the attachments using the field name as reference.
OLE Flag as work item field

There is a possibility to get an information into a separate work item field, whenever the work item contains OLE objects to easily find those items by a work item query.

The attribute ‘OLEMarkerField’ can be set to an existing field which has to be available in the work item type definition or custom field. If it is a custom field, the TFS administrator needs to make sure the field has been added to the work item type definition. Field type should be plain text. The attribute ‘OLEMarkerValue’ can be set to custom value.

Both attributes "OLEMarkerField" and "OLEMarkerValue" are optional. If "OLEMarkerField" is not defined, the behavior is like the old one, which means nothing is done. If "OLEMarkerField" is defined, but "OLEMarkerValue" is not defined, the default value for "OLEMarkerValue" is taken, which is the field name of the appropriate work item field.

Example for OLE Flag using an excel file in description field

```xml
<Field Name="System.Description"
    FieldValueType="HTML"
    Direction="OtherToTfs"
    MappingTableRow="3"
    MappingTableCol="1"
    HandleAsDocument="true"
    HandleAsDocumentMode="OleOnDemand"
    OLEMarkerField="Custom.MyFields.DescriptionOLE"
    OLEMarkerValue="Y">...
</Field>
```

From a technical point of view excel file in Word are OLE objects.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Test OLE Marker Flag</th>
<th>314</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Functional</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Administrator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After publishing the work item to TFS, the custom field with reference name ‘Custom.MyFields.DescriptionOLE’ will be set to ‘Y’.

<table>
<thead>
<tr>
<th>CLASSIFICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
</tr>
<tr>
<td>Reaction</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Value area</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OLE CUSTOM FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DescriptionOLE</td>
</tr>
<tr>
<td>AnalysisOLE</td>
</tr>
<tr>
<td>OLEAvailable</td>
</tr>
</tbody>
</table>
In case when two work items are mapped to one field on TFS, then both marker values will be added to the same field separated by comma. If both marker values are same than just one value will be set.

**Converters/Converter**

A converter can help you to show the values in the right way.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>TFS field name to convert</td>
</tr>
<tr>
<td>Text</td>
<td>TFS Value</td>
</tr>
<tr>
<td>MappingText</td>
<td>Word Value</td>
</tr>
</tbody>
</table>

**Example:**

```xml
<Converters>
  <Converter FieldName="Microsoft.VSTS.Common.Severity">
    <Values>
      <Value Text="1 - Critical" MappingText="1" />
      <Value Text="2 - High" MappingText="2" />
      <Value Text="3 - Medium" MappingText="3" />
      <Value Text="4 - Low" MappingText="4" />
      <Value Text="4 - Low" MappingText="" />
    </Values>
  </Converter>
</Converters>
```

**Defining link mappings**

Using the links element and the link element, you can set up table cells to contain formatted work item links.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Name of the relation. For example “child” if you want to edit children in this field, “parent” if you want to edit the parent. You can use any registered link type.</td>
</tr>
<tr>
<td>Direction</td>
<td>Synchronization direction. See table</td>
</tr>
<tr>
<td>MappingTableRow</td>
<td>Row of the field within the template.</td>
</tr>
<tr>
<td>MappingTableCol</td>
<td>Column of the field within the template.</td>
</tr>
<tr>
<td>Overwrite</td>
<td>Sets whether the links in the mapped cell overwrite existing links. “False” means, links can only be added. “True” means, links deleted in Word are also deleted on your TFS. This attribute is only meaningful when combined with directions “OtherToTfs” or “PublishOnly”.</td>
</tr>
<tr>
<td>LinkSeparator</td>
<td>Character sequence used to separate multiple entries. Use \t or \r for tab or return respectively.</td>
</tr>
<tr>
<td>LinkFormat</td>
<td>A composite format string used to format output for a single link. You can access any field of the referenced work item by “{FieldName}”</td>
</tr>
</tbody>
</table>
LinkedWorkItemTypes | List of comma separated values to specify the work item type(s) to be linked. When not specified, all work item types are included.

AutoLinkWorkItemType | Setting this to a valid work item type will automatically add a link to the last work item of the given type that appears before the current work item in the Word document.

AutoLinkWorkItemSubtypeField | Setting this to a valid field reference name will automatically add a link to the last work item of type “AutoLinkWorkItemType” that appears before the current work item in the Word document and whose “AutoLinkWorkItemSubtypeField” has the value “AutoLinkWorkItemSubtypeValue”.

AutoLinkWorkItemSubtypeValue | Setting this to a valid value will automatically add a link to the last work item of type “AutoLinkWorkItemType” that appears before the current work item in the Word document and whose “AutoLinkWorkItemSubtypeField” has the value “AutoLinkWorkItemSubtypeValue”.

AutoLink SuppressWarnings | Setting this to “False” will suppress warnings if automatic linking was defined but no appropriate work item was found in the document.

<table>
<thead>
<tr>
<th>Get</th>
<th>Refresh</th>
<th>Publish</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetOnly</td>
<td>Work item links are refreshed</td>
<td>-</td>
</tr>
<tr>
<td>TfsToOther</td>
<td>Work item links are refreshed</td>
<td>Work item links are refreshed</td>
</tr>
<tr>
<td>OtherToTfs</td>
<td>Work item links are refreshed</td>
<td>Work item links are refreshed</td>
</tr>
<tr>
<td>PublishOnly</td>
<td>Work item links are refreshed</td>
<td>-</td>
</tr>
<tr>
<td>SetInNewTfsWorkItem</td>
<td>Work item links are refreshed</td>
<td>Work item links are refreshed</td>
</tr>
</tbody>
</table>

Example 1
```xml
<Mapping>
  <Fields>
  ...
  </Fields>
  <Links>
    <Link Type="Child"
      LinkedWorkItemTypes="Bug, Task"
      Direction="OtherToTfs"
      MappingTableRow="4"
      MappingTableCol="1"
      Overwrite="false"
      LinkFormat="{System.Id} – {System.Title}" />
  </Links>
</Mapping>
```
This example configures the fourth cell in the first row of the table to display all links to all children which have the work item type *Bug* or *Task* in the format “1929 – Title”. If the work item has multiple children, each entry is displayed on a new line. Changes made in Word will be published, but only if they add children. Deleting children will not be possible.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Example Requirement</th>
<th>250</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This is the <strong>description</strong> of the example requirement. 272 – Title of the child bug work item</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example 2**

```xml
<Links>
  ...
  <Link Type="Parent"
    Direction="OtherToTfs"
    MappingTableRow="5"
    MappingTableCol="1"
    Overwrite="true"
    AutoLinkWorkItemType="Requirement"
    AutoLinkWorkItemSubtypeField="Microsoft.VSTS.CMMI.RequirementType"
    AutoLinkWorkItemSubtypeValue="Business Objective"
    LinkFormat="{System.Id} - {System.Title}" />
</Links>
```

This example extends the previous one and configures the cell in the fourth row and the second column of the table to display a link to the parent work item. When publishing, WordToTFS searches from the current work item to the top of the document for the first Requirement of type Business Objective and links to that work item.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Example Requirement</th>
<th>250</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This is the <strong>description</strong> of the example requirement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parent:</td>
<td>274 – Example Feature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Children:</td>
<td>272 – Title of the child bug work item</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Example Feature</th>
<th>274</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator</td>
<td>Proposed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bug</th>
<th>Title of the child bug work item</th>
<th>272</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort:</td>
<td>3 · Medium</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks**

Be aware of the following restrictions:

- The link format must always contain the field reference “{System.Id}”.

AIAIT - WordToTFS User Guide – V6.3.0.0
• The link separator must not be contained in either the link format string or the referenced fields. For example using a comma as link separator will not work if one of the referenced fields contains a comma as well.

• When referencing multiple fields like System.Id and System.Title in the example, make sure that the character sequence between the referenced fields does not appear in either of the separated field values. The above example would not work if the System.Title contains the sequence "–".

**Nodes <Transitions>, <Transition>**

This elements can be used to configure an automatic state transition during publish.

**Attribute for <Transitions> and <Transition>**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>Reference name of the field that should be altered.</td>
</tr>
<tr>
<td>From</td>
<td>Identifier of the current state</td>
</tr>
<tr>
<td>To</td>
<td>Identifier of the next state</td>
</tr>
</tbody>
</table>

In some cases, fields are read-only for a work item in its current state. If by changing these fields in Word you mean to imply that the state of the work item also changes, you can use automatic state transitions to define what state to transition to:

**Example:**

```xml
<Transitions FieldName="System.State">
  <Transition From="Released" To="Draft" />
  <Transition From="Approved" To="Draft" />
</Transitions>
```

This example shows how to reset the state field of a work item back to “Draft” if the user makes changes to the mapped work item. The referenced state field (here “System.State”) does not have to be configured using the <Field> element, but if it is then:

1. The value changed in Word, overrides the transition rule (the user value wins)
2. The Direction of the field must be set to “OtherToTfs”

**Defining Headers**

Headers are defined within <Header></Header> tags in the MappingConfiguration/Headers section of the configuration file:

```xml
<MappingConfiguration>
  <Headers>
    <Header Identifier ... />
  </Headers>
  <Mappings>
  </Mappings>
</MappingConfiguration>
```

The <Header>-Element has the following Attributes:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Name of the header.</td>
</tr>
<tr>
<td>Row</td>
<td>Row of the cell in which to look for the identifier</td>
</tr>
<tr>
<td>Column</td>
<td>Column of the cell in which to look for the identifier</td>
</tr>
<tr>
<td>Level</td>
<td>The hierarchy level of the header. Field definitions of a header are overwritten by headers of higher level (within their range) or by work items if mapped. A header will affect work items until another header of the same or lower level appears in the text.</td>
</tr>
<tr>
<td>RelatedTemplate</td>
<td>As in &lt;Mapping&gt;</td>
</tr>
<tr>
<td>ImageFile</td>
<td>As in &lt;Mapping&gt;</td>
</tr>
</tbody>
</table>

Within the <Header>-Tag you can define Field mappings as for normal work item mappings. You can also define converters, but you cannot define state transitions or links. The only supported directions for fields in headers are `SetInNewTfsWorkItem` and `WordToTFS`.

If the same field is mapped by multiple headers within the same range, only the last header definition applies. If the same field is mapped by a header and an affected work item, publishing result depends on the field direction. If it is set to `WordToTFS`, the value is always read from the header and the header configuration applies. This will overwrite changes made to individual affected work items. If an affected field is changed from outside word, the usual conflict checking mechanism applies. If the header value conflicts with a newer value of an affected work item, you have to refresh the work item first, but the changes will then be overwritten with the header field.

If the direction is set to `SetInNewTfsWorkItem`, the value is read from the header and the configuration of the header applies for all affected work items where no value is set. If any value is set in the work item, the work item configuration applies and the value of the work item is published. In any case, all subsequent operations will use the work item field configuration.

4: Example header configuration

```xml
<Header Identifier="DocumentHeader"
    Level="1" Row="1" Column="1"
    RelatedTemplate="TeamSystemPro(2010).DocumentHeader.xml"
    ImageFile="standard.png">
    <Fields>
        <Field Name="Microsoft.VSTS.Common.Priority"
            Direction="OtherToTfs"
            FieldValueType="PlainText"
            MappingTableRow="2"
            MappingTableCol="2" />
        <Field Name="System.AssignedTo"
            Direction="OtherToTfs"
            FieldValueType="PlainText"
            MappingTableRow="3"
            MappingTableCol="2" />
    </Fields>
    <Converters>
        <Converter FieldName="Microsoft.VSTS.Common.Priority">
            <Values>
                <Value Text="1" MappingText="Critical" />
                <Value Text="2" MappingText="High" />
                <Value Text="3" MappingText="Medium" />
                <Value Text="4" MappingText="Low" />
            </Values>
        </Converter>
    </Converters>
</Header>
```
Defining Variables

Variables are defined within `<Variables>` tags underneath the MappingConfiguration of the configuration file:

```xml
<MappingConfiguration>
  <Variables>
    <Variable Name="Replace" Value="InsertText"/>
  </Variables>
</MappingConfiguration>
```

These variables can then be used to insert additional text in tables (cf.).

Collapsing/Expanding of the query tree

By default, the tree view showing work item queries is expanded. If you have a lot of queries you might not find it easy to navigate through the complete tree structure. Therefore, it is possible to add the attribute `CollapsQueryTree="true"` to show a collapsed tree view initially. This configuration affects both occurrences: “Get Work Items” and “Test Specification Report By Query”.

```xml
<MappingConfiguration xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="W2T.xsd"
  ShowName="CMMI(2015)"
  ...
  CollapsQueryTree="true">
  ...
</MappingConfiguration>
```

Customizing the template layout: .xml files

The Layout and design is provided through table xml files. To generate these files, just create a blank word file with a table, and save it as “Word Xml File”. The simplest work item table is a simple table with some rows and cols. But you also can use background color, merge cells (only horizontally), invisible borders, font size, type and color and so on. Save them in the same folder as your other template stuff, the w2t file and the optional favicon, and generate the entry for the file at the w2t mapping definition. Ensure that the filename is written right at the `RelatedTemplate` – Tag at the Mapping – Node.

Additional you have to make sure you have the right amount of cells. As example, your word file contains a table with 5 columns and 2 rows as shown below, you can’t map a TFS field to col 1 with row 3.

<table>
<thead>
<tr>
<th>row 1, col 1</th>
<th>row 1, col 2</th>
<th>row 1, col 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 2, col 2</td>
<td>row 2, col 3</td>
<td>row 2, col 6</td>
</tr>
<tr>
<td>Nonexisting cell</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Configuring the test reporting extension
WordToTFS supports generating test specification reports and test result reports. The specification report is intended to generate an overview of how the tests were conducted while the result report gives an overview of the latest test outcomes. The general structure of the generated document is fixed, but the individual components can be customized independently for both report types. Typically more verbose information like test setup is shown in the specification report and the result report only lists the test case title and id for a test case along with its test outcomes. It is of course possible to use the same component configuration in both report types.

The basic configuration looks like this:

```xml
<MappingConfiguration>
  <TestConfiguration>
    <TestSpecificationConfiguration ... />
    <TestResultConfiguration ... />
    <Templates>
      ...
    </Templates>
  </TestConfiguration>
</MappingConfiguration>
```

The `TestResultConfiguration` and `TestSpecificationConfiguration` define what templates are used for the various components that make up a report. The templates themselves are defined within the `Templates` section.

**General test configuration**

```xml
<TestConfiguration setHyperlinkBase="true" ExpandSharedSteps="true">
  <Attribute Explanation>
    SetHyperlinkBase (true|false) When enabled, the document hyperlink base is set to the current document location. Turn this on if you use attachment downloading but your hyperlinks do not work. Hyperlinks will stop working for word installations without WordToTFS plugin after the document is moved to a different location if this option is turned on.
  </Attribute>
  <Attribute Explanation>
    ExpandSharedSteps (true|false) When enabled, the shared steps in test reporting will be expanded and minor step numbers will be used to show steps belonging to the shared step. The default value is false and shared steps will not be expanded.
  </Attribute>
</TestConfiguration>
```

**Test specification report configuration**

```xml
<TestSpecificationConfiguration>
  Available="true"
  TestPlanTemplate="TestPlan"
  TestSuiteTemplate="TestSuite"
  LeafTestSuiteTemplate="LeafTestSuite"
  RootTestSuiteTemplate="RootTestSuite"
  TestCaseElementTemplate="SpecificationReportTestSuiteRow"
  SummaryPageTemplate="OwnSummaryPage" 
  <PreOperations>
    <Operation Type="InsertNewPage"/>
  </PreOperations>
  <PostOperations>
    <Operation Type="RefreshAllFieldsInDocument"/>
  </PostOperations>
  <DefaultValue>
    ...
  </DefaultValue>
</TestSpecificationConfiguration>
```
### Attribute | Explanation
--- | ---
**Available (true|false)** | Shows (true) or hides (false) the button in the word ribbon, thereby allowing or preventing users from generating test specification reports.

**TestPlanTemplate** | Name of a template configured in the templates section that is used to generate test plan information. If this attribute is missing no test plan information is shown in the report.

**TestSuiteTemplate** | Name of a template configured in the templates section that is used to generate test suite information. If this attribute is missing no test suite information is shown in the report.

**LeafTestSuiteTemplate** | Name of a template configured in the templates section that is used to generate test suite information for leaf test suites. If this attribute is missing the TestSuiteTemplate is used. Further if a leaf test suite is also the root test suite the RootTestSuiteTemplate is applied.

**RootTestSuiteTemplate** | Name of a template configured in the templates section that is used to generate test suite information for the test suite selected as root for the report. If this attribute is missing the TestSuiteTemplate is used.

**SummaryPageTemplate** | Name of a template configured in the templates section that is used to generate a summary page at the end of the report. If this attribute is missing no summary page is shown.

**PreOperations** | The definition of an array of operation that should be performed prior to insertion of the template / after inserting the template to target document. The Operation is performed in target document.

**PostOperations** | The definition of an array of operation that should be performed prior to insertion of the template / after inserting the template to target document. The Operation is performed in target document.

**Operation** | Type of the operation defines the operation to execute. Possible types:
- InsertParagraph
- MoveCursorToStart
- MoveCursorToEnd
- DeleteCharacterLeft
- DeleteCharacterRight
- MoveCursorToLeft
- MoveCursorToRight
- InsertNewPage
- RefreshAllFieldsInDocument

**DefaultValues** | Sets the default values for the test reporting configuration. The according UI elements will be set to the configured values, if the assignment is possible. If not, a fallback to standard value will be performed.

---

**Test result report configuration**

<TestResultConfiguration
Available="true"
```
TestPlanTemplate="TestPlan"
TestSuiteTemplate="TestSuite"
LeafTestSuiteTemplate="LeafTestSuite"
RootTestSuiteTemplate="RootTestSuite"
TestCaseElementTemplate="ResultReportTestCaseRow"
SharedStepsElementTemplate="STSSharedSteps"
TestResultElementTemplate="ResultReportTestResultRow"
TestConfigurationElementTemplate="ResultReportTestConfigurationRow"
SummaryPageTemplate="OwnSummaryPage"

<PreOperations>
    <Operation Type="InsertNewPage"/>
</PreOperations>

<PostOperations>
    <Operation Type="RefreshAllFieldsInDocument"/>
</PostOperations>

<DefaultValues>
    ...
</DefaultValues>
</TestResultConfiguration>
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available (true</td>
<td>false)</td>
</tr>
<tr>
<td>TestPlanTemplate</td>
<td>Name of a template configured in the templates section that is used to generate test plan information. If this attribute is missing no test plan information is shown in the report.</td>
</tr>
<tr>
<td>TestSuiteTemplate</td>
<td>Name of a template configured in the templates section that is used to generate test suite information. If this attribute is missing no test suite information is shown in the report.</td>
</tr>
<tr>
<td>LeafTestSuiteTemplate</td>
<td>Name of a template configured in the templates section that is used to generate test suite information for leaf test suites. If this attribute is missing the TestSuiteTemplate is used. Further if a leaf test suite is also the root test suite the RootTestSuiteTemplate is applied.</td>
</tr>
<tr>
<td>RootTestSuiteTemplate</td>
<td>Name of a template configured in the templates section that is used to generate test suite information for the test suite selected as root for the report. If this attribute is missing the TestSuiteTemplate is used.</td>
</tr>
<tr>
<td>TestCaseElementTemplate</td>
<td>Name of a template configured in the templates section that is used to generate test case information. If this attribute is missing no test case information is shown in the report.</td>
</tr>
<tr>
<td>SharedStepsElementTemplate</td>
<td>The name of a template configured in the templates section that is used to generate shared step information. If this attribute is missing no shared step information is shown in the report.</td>
</tr>
<tr>
<td>TestResultElementTemplate</td>
<td>Name of a template configured in the templates section that is used to generate test result information. If this attribute is missing no test result information is shown in the report.</td>
</tr>
</tbody>
</table>
TestConfigurationElementTemplate | Name of a template configured in the templates section that is used to generate test configuration information. If this attribute is missing no test configuration information is shown in the report.

SummaryPageTemplate | Name of a template configured in the templates section that is used to generate a summary page at the end of the report. If this attribute is missing no summary page is shown.

PreOperations
PostOperations | The definition of an array of operation that should be performed prior to insertion of the template / after inserting the template to target document. The Operation is performed in target document.

Operation | Type of the operation defines the operation to execute. Possible types:
- InsertParagraph
- MoveCursorToStart
- MoveCursorToEnd
- DeleteCharacterLeft
- DeleteCharacterRight
- MoveCursorToLeft
- MoveCursorToRight
- InsertNewPage
- RefreshAllFieldsInDocument

DefaultValues | Sets the default values for the test reporting configuration. The according UI elements will be set to the configured values, if the assignment is possible. If not, a fallback to standard value will be performed.

Test run result

```xml
<Template Name="ResultReportTestResultRow"
  HeaderTemplate="ResultReportTestResultHeader"
  FileName="MSFForCMMI.Test.Result.TestResult.Row.xml">
  <PreOperations>
    <Operation Type="DeleteCharacterLeft"/>
  </PreOperations>
  <Replacements>
    <Replacement Bookmark="RunId" Property="TestRunId"/>
    <Replacement Bookmark="Outcome" Property="Outcome"/>
    <Replacement Bookmark="Duration" Property="Duration"/>
    <Replacement Bookmark="State" Property="State"/>
    <Replacement Bookmark="RunBy" Property="RunBy"/>
    <Replacement Bookmark="DateCompleted" Property="DateCompleted"/>
    <Replacement Bookmark="ConfigID" Property="TestConfigurationId"/>
    <Replacement Bookmark="ConfigName" Property="TestConfigurationName"/>
    <Replacement Bookmark="BuildNumber" Property="BuildNumber"/>
    <BuildViewerLink BuildNumber="BuildNumber"/>
  </Replacements>
</Template>
```
### Replacements

<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunId</td>
<td>Identification number of the test run.</td>
</tr>
<tr>
<td>Outcome</td>
<td>Shows the result of the execution of the test. Available: Passed</td>
</tr>
<tr>
<td>Duration</td>
<td>Shows the period required to complete the execution of the test.</td>
</tr>
<tr>
<td>State</td>
<td>Shows the execution state of the test. Available: Completed</td>
</tr>
<tr>
<td>RunBy</td>
<td>Name of the user who run the test.</td>
</tr>
<tr>
<td>DateCompleted</td>
<td>Shows the end date of test execution.</td>
</tr>
<tr>
<td>ConfigID</td>
<td>Shows identification number of the test configuration.</td>
</tr>
<tr>
<td>ConfigName</td>
<td>Shows name of the test configuration.</td>
</tr>
<tr>
<td>BuildNumber</td>
<td>Identification number of build.</td>
</tr>
<tr>
<td>Comments</td>
<td>Shows content of description for analysis why a test failed.</td>
</tr>
<tr>
<td>OwnerName</td>
<td>Name of the user that created test.</td>
</tr>
<tr>
<td>FailureType</td>
<td>The type of failure. Available: None</td>
</tr>
<tr>
<td>ResolutionStateId</td>
<td>Specifies the reason why a test failed. Available: None</td>
</tr>
</tbody>
</table>

This replacement contain attribute `ResolveResolutionState` (true | false ) that determines the display format of `ResolutionState`. If it is true , `ResolutionState` is displayed as a string and if it is false, `ResolutionState` is displayed as a number.

### Default values for test reporting

Available default values for test specification report

```
<DefaultValues>
  <SelectTestSuite>Suite2.1</SelectTestSuite>
  <SelectTestPlan>TestPlan1</SelectTestPlan>
  <!--TestPlanHierarchy|AreaPath|IterationPath-->
  <CreateDocumentStructure>false</CreateDocumentStructure>
  <DocumentStructureType>TestPlanHierarchy</DocumentStructureType>
  <SkipLevels>3</SkipLevels>
  <!--AboveTestPlan|BeneathTestPlan|BeneathTestSuites|BeneathFirstTestSuite-->
  <IncludeTestConfigurations>false</IncludeTestConfigurations>
  <ConfigurationPositionType>BeneathTestSuites</ConfigurationPositionType>
  <!--None|IterationPath|AreaPath|WorkItemId-->
  <SortTestCasesBy>WorkItemId</SortTestCasesBy>
</DefaultValues>
```

<table>
<thead>
<tr>
<th>XML Node</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectTestSuite</td>
<td>The name of the test suite that will be selected.</td>
</tr>
<tr>
<td>SelectTestPlan</td>
<td>The name of the test plan template that will be selected.</td>
</tr>
<tr>
<td>CreateDocumentStructure (true/false)</td>
<td>Should the document structure be used, for report generation</td>
</tr>
</tbody>
</table>
**DocumentStructureType**  | The document structure type that will be selected. Available: TestPlanHierarchy | AreaPath | IterationPath
---|---
**SkipLevels**  | The number of levels that will be skipped.
**IncludeTestConfigurations** *(true/false)*  | Defines if the test configuration will be included
**ConfigurationPositionType**  | Defines the selected configuration position type. Available: AboveTestPlan | BeneathTestPlan | BeneathTestSuites | BeneathFirstTestSuite
**SortTestCasesBy**  | Defines the sort type. Available: None | IterationPath | AreaPath | WorkItemId

### Available default values for test result report

```xml
<DefaultValues>
    <!--TestPlanHierarchy|AreaPath|IterationPath-->
    <CreateDocumentStructure>true</CreateDocumentStructure>
    <DocumentStructureType>TestPlanHierarchy</DocumentStructureType>
    <SkipLevels>3</SkipLevels>
    <!--AboveTestPlan|BeneathTestPlan|BeneathTestSuites|BeneathFirstTestSuite-->
    <IncludeTestConfigurations>true</IncludeTestConfigurations>
    <ConfigurationPositionType>BeneathTestSuites</ConfigurationPositionType>
    <!--None|IterationPath|AreaPath|WorkItemId-->
    <SortTestCasesBy>WorkItemId</SortTestCasesBy>
    <IncludeMostRecentTestResult>true</IncludeMostRecentTestResult>
    <IncludeMostRecentTestResultForAllSelectedConfigurations>true</IncludeMostRecentTestResultForAllSelectedConfigurations>
    <SelectTestPlan>TestPlan1</SelectTestPlan>
    <SelectTestConfiguration>Windows 7</SelectTestConfiguration>
    <SelectBuild>ConsoleApplication1 B1_20151005.3</SelectBuild>
</DefaultValues>
```

<table>
<thead>
<tr>
<th><strong>XML Node</strong></th>
<th><strong>Explanation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectTestPlan</td>
<td>The name of the test plan template that will be selected.</td>
</tr>
<tr>
<td>CreateDocumentStructure <em>(true/false)</em></td>
<td>Should the document structure be used, for report generation</td>
</tr>
<tr>
<td>DocumentStructureType</td>
<td>The document structure type that will be selected. Available: TestPlanHierarchy</td>
</tr>
<tr>
<td>SkipLevels</td>
<td>The number of levels that will be skipped.</td>
</tr>
<tr>
<td>IncludeTestConfigurations <em>(true/false)</em></td>
<td>Defines if the test configuration will be included</td>
</tr>
<tr>
<td>ConfigurationPositionType</td>
<td>Defines the selected configuration position type. Available: AboveTestPlan</td>
</tr>
<tr>
<td>SortTestCasesBy</td>
<td>Defines the sort type. Available: None</td>
</tr>
<tr>
<td>IncludeMostRecentTestResult <em>(true/false)</em></td>
<td>Defines, if the most recent test results should be included.</td>
</tr>
<tr>
<td>IncludeMostRecentTestResultForAllSelectedConfigurations <em>(true/false)</em></td>
<td>Defines, if the most recent test results should be included for all selected configurations</td>
</tr>
<tr>
<td>SelectTestConfiguration</td>
<td>Defines the selected configuration.</td>
</tr>
<tr>
<td>SelectBuild</td>
<td>Defines the selected build. Note: Build must be configured and run within test context.</td>
</tr>
</tbody>
</table>
Identify test suite selection by tree path

On different places, it is possible to define the test suite which should be selected for the test report generation. For example, this can be defined in the default settings or within specific settings for the WordToTFS console. If a test plan has multiple hierarchy-levels and maybe even names are used multiple times, it might not be enough to identify the test suite just by name and search within the tree structure. Therefore, it is possible to use a tree path to identify the test suite which should be used for selection.

Example 1: Default Values

```
<DefaultValues>
  <SelectTestSuite>TestPlan1RootSuite/SubSuite1/SubSub2</SelectTestSuite>
  <SelectTestPlan>TestPlan1</SelectTestPlan>
  <!-- TestPlanHierarchy/AreaPath/IterationPath-->
  <CreateDocumentStructure>false</CreateDocumentStructure>
  <DocumentStructureType>TestPlanHierarchy</DocumentStructureType>
  <SkipLevels>3</SkipLevels>
  <!-- AboveTestPlan|BeneathTestPlan|BeneathTestSuites|BeneathFirstTestSuite-->
  <IncludeTestConfigurations>false</IncludeTestConfigurations>
  <ConfigurationPositionType>BeneathTestSuites</ConfigurationPositionType>
  <!-- None|IterationPath|AreaPath|WorkItemId-->
  <SortTestCasesBy>WorkItemId</SortTestCasesBy>
</DefaultValues>
```

Example 2: Specific Settings for Console Test Specification Settings

```
<TestConfiguration>
  <TestSpecificationConfiguration>
    TestPlan="MyPlan"
    TestSuite="TestPlan1RootSuite/SubSuite1/SubSub2"
    CreateDocumentStructure="true"
    DocumentStructureType="TestPlanHierarchy"
    SkipLevels="2"
    IncludeTestConfigurations="true"
    TestConfigurationsPosition="AboveTestPlan"
    SortTestCasesBy="WorkItemId" /
  </TestSpecificationConfiguration>
</TestConfiguration>
```

Templates

A template defines a view on an object of the tfs object model. It consists of the actual view that is an external word document (typically a table) and information on how to bind the view to the object properties. The external document is inserted into the generated report whenever information of the object the view is defined for should be reported. The inserted document is then searched for bookmarks and each bookmark is replaced with the value of a property as configured in the template.

```
<Template Name="TestPlan" HeaderTemplate="" FileName="TestPlan.xml">
  <PreOperations>
    <Operation Type="DeleteCharacterLeft"/>
  </PreOperations>
  <PostOperations>
    <Operation Type="MoveCursorToLeft"/>
  </PostOperations>
  <Replacements>
```

AIT - WordToTFS User Guide – V6.3.0.0
This example configuration instructs the generation algorithm to insert the file “TestPlan.xml” into the document when the document structure requires showing test plan information. It then tries to replace the bookmark “Name” with the name of the test plan, the bookmark “AreaPath” with the area path of the test plan etc.

<table>
<thead>
<tr>
<th>Attribute/Node</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>Node for the definition of one node.</td>
</tr>
<tr>
<td>Name</td>
<td>User defined unique name that is used to identify the template.</td>
</tr>
<tr>
<td>HeaderTemplate</td>
<td>Name of template to use as header. The name is used only in such case, if the actual template is used to generate a block of the elements of the same type. This feature is usable to generate a table with header.</td>
</tr>
<tr>
<td>FileName</td>
<td>The name of the file used as Word template inserted on the required location in generated report. (The structure of generated report is described in chapter “Select Builds – Filtering based on build association This option is only available for XAML-based builds. The new build Engine concept introduced with TFS 2015 is not supported. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build quality assigned, which is configured in the WordToTFS-template. Please check the WordToTFS-template “CMMI(2015)” for details.</td>
</tr>
</tbody>
</table>

```xml
<MappingConfiguration>
  <TestConfiguration>
    <TestSpecificationConfiguration ... />
    <TestResultConfiguration>
      <BuildQualities>
        <BuildQuality>Released</BuildQuality>
        <BuildQuality>Passed</BuildQuality>
        <BuildQuality>Ready</BuildQuality>
      </BuildQualities>
    </TestResultConfiguration>
  </TestConfiguration>
</MappingConfiguration>
```

Select Builds – Filtering based on build-filters
This option is available for all builds. In this selection field, not all builds will be displayed, only those which are within the current connected project and have the certain build filter assigned, which is configured in the WordToTFS-template. If <BuildFilters> exist in template, then separated <BuildQualities> is not allowed. <BuildAge> represent the age of build relative to today,
e.g. if value is 21, only builds younger than 21 days will be shown. Filters <BuildName> and <BuildAge> are available for all builds, until <BuildQualities> is only available for XAML-based builds and <BuildTags> is only available for vNext builds.

```xml
<MappingConfiguration>
  <TestConfiguration>
    <TestSpecificationConfiguration ... />
    <TestResultConfiguration>
      <BuildFilters>
        <BuildAge>21</BuildAge>
        <BuildNames>
          <BuildName>Name of XAML build</BuildName>
          <BuildName>Name of vNext build</BuildName>
        </BuildNames>
        <BuildTags>
          <BuildTag>Ready</BuildTag>
          <BuildTag>Test</BuildTag>
        </BuildTags>
        <BuildQualities>
          <BuildQuality>Released</BuildQuality>
          <BuildQuality>Passed</BuildQuality>
          <BuildQuality>Ready</BuildQuality>
        </BuildQualities>
      </BuildFilters>
    </TestResultConfiguration>
  </TestConfiguration>
</MappingConfiguration>
```

Structure of generated report”.

<table>
<thead>
<tr>
<th>Replacement</th>
<th>The definition for a single replacement. The bookmark in inserted template will be replaced by value of evaluated property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookmark</td>
<td>Name of the bookmark to replace.</td>
</tr>
<tr>
<td>Property</td>
<td>The property of the bound object that replaces the bookmark. The property can be nested – “RootSuite.Name”. If the path contains a string indexer, the key must be enclosed in double quotation marks: ‘CustomFields[“AIT.Description”].Value’</td>
</tr>
<tr>
<td>ValueType</td>
<td>Type of the value of the evaluated property. Possible values are “PlainText” and “HTML”. If this attribute is omitted, the value “PlainText” is used.</td>
</tr>
<tr>
<td>PreOperations</td>
<td>The definition of an array of operation that should be performed prior to insertion of the template, after inserting the template to target document. The Operation is performed in target document.</td>
</tr>
<tr>
<td>PostOperations</td>
<td>Type of the operation defines the operation to execute. Possible types:</td>
</tr>
<tr>
<td></td>
<td>• InsertParagraph</td>
</tr>
<tr>
<td></td>
<td>• MoveCursortToStart</td>
</tr>
<tr>
<td></td>
<td>• MoveCursortToEnd</td>
</tr>
<tr>
<td></td>
<td>• DeleteCharacterLeft</td>
</tr>
<tr>
<td></td>
<td>• DeleteCharacterRight</td>
</tr>
<tr>
<td></td>
<td>• MoveCursorToLeft</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>• MoveCursorToRight</td>
<td></td>
</tr>
<tr>
<td>• InsertNewPage</td>
<td></td>
</tr>
<tr>
<td>• RefreshAllFieldsInDocument</td>
<td></td>
</tr>
</tbody>
</table>
Replacement with variables within templates

For replacements in the test reporting templates also variables can be used as mentioned in the previous section Defining Variables. Therefore, the replacement needs to be set to either FieldValueTyp=BasedOnVariable or FieldValueTyp=BasedOnSystemVariable.

```xml
<Variables>
  <Variable Name="Replace1" Value="InsertText"/>
</Variables>

[...]
<Template Name="TestPlan" HeaderTemplate="" FileName="TestPlan.xml">
  [...]
  <Replacements>
    <Replacement Bookmark="Name" Property="Name"/>
    <Replacement Bookmark="AreaPath" Property="AreaPath"/>
    <Replacement Bookmark="MyVariableBookmark"
        FieldValueTyp=BasedOnVariable VariableName="Replace1"/>
    <Replacement Bookmark="MySystemVariableBookmark"
        FieldValueTyp=BasedOnSystemVariable VariableName="WordToTFS.Version"/>
  </Replacements>
</Template>

Invoke methods from within the template

In addition to bind bookmarks to properties, it is also possible to invoke a method on the property before the value is inserted into the document. If you want a custom date format, you can invoke the ToString-Method and pass the formatting string in parentheses. Note that you have to use single quotation marks for the property if you want to pass a string argument to the method.

```xml
<Template ... >
  <Replacements>
    <Replacement Bookmark="Date" Property='Date.ToString("dd-mmm-YYYY")'/>
  </Replacements>
</Template>

Object used to resolve property value

We have defined that 4 templates for Test specification report can be defined and 8 templates for Test result report. For every template is dedicated one object from Microsoft object model in Team Foundation Server. Therefore the user must look at the corresponding documentation, if he wants to define the properties that need to be evaluated in replacement phase of creating of report.

Table 5: Base objects used in replacement phase of generation of report

<table>
<thead>
<tr>
<th>Template</th>
<th>Interface from object model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test plan</td>
<td>ITestPlan (Link to interface on MSDN)</td>
</tr>
<tr>
<td>Test suite</td>
<td>ITestSuiteBase (Link to interface on MSDN)</td>
</tr>
<tr>
<td>Test case</td>
<td>ITestCase (Link to interface on MSDN)</td>
</tr>
<tr>
<td>Test result</td>
<td>ITestCaseResult (Link to interface on MSDN)</td>
</tr>
<tr>
<td>Test configuration</td>
<td>ITestConfiguration (Link to interface on MSDN)</td>
</tr>
</tbody>
</table>

Examples of properties in ITestCase:

- WorkItem.Id – identification of the test case
- WorkItem.Rev – revision of the test case
- WorkItem.Title – title of the test case
Actions – actions / steps in test case – formatted output of array is described in chapter “Hyperlink to viewer of test attachments”

The occurrence of an attachment link will trigger downloading the attachment to a subfolder of the document for the modes “DownloadOnly” and “DownloadAndLinkToLocalFile”. The properties “LocalPath” and “LocalFilename” are only available using these modes.

Using the mode “DownloadOnly” in combination with a property other than “LocalPath” or “LocalFilename” is not recommended as the attachments will be downloaded but there is usually no way to tell which a attachments belongs to which occurrence in the document.

In case of using the mode “DownloadAndLinkToLocalFile” there are three options. The attachment folder can be named with a guid, without a guid or based on test suite. This has to be specified in the MappingConfigurations of the w2t file using the property AttachmentFolderMode.

Possible values are WithGuid, WithoutGuid and BasedOnTestSuite, whereas default is the first one.

- WithGuid – Name of attachment folder contains GUID. Name of attachment file contains GUID.
- WithoutGuid - Name of attachment folder doesn’t contain GUID. Name of attachment file contains GUID.
- BasedOnTestSuite – For every test suite exists folder with name and ID of test suite. Inside that test suite folder can be a new folder which represents child test suite or attachment file which name contains name and ID of test case. So, there exists test suite hierarchy with n-depth.

File and folder structure is the following:

- File XYZ.docx (Test Report)
- Folder “TS A (%Test Suite ID)”
  - File “TC A.1 (%Test Case ID%)_0.png”
  - File “TC A.1 (%Test Case ID%)_1.png”
  - File “TC A.1 (%Test Case ID%)_2.jpeg”
- Folder “TS C (%Test Suite ID)”
  - Folder “TS C_II (%Test Suite ID)”
    - File “TC C_II.1 (%Test Case ID)_0.png”

```xml
<MappingConfiguration xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="W2T.xsd"
  ShowName="CMMI(2015)"
  AttachmentFolderMode="BasedOnTestSuite">
  ...
  <Template Bookmark="AttachmentSimpleList" Property="...">
    <Replacements>
      <Replacement Bookmark="Name" Property="Name">
        <AttachmentLink Mode="DownloadAndLinkToLocalFile"/>
      </Replacement>
    </Replacements>
  </Template>
</MappingConfiguration>
```
Object model extensions
In addition to the properties found in the MSDN documentation, some objects are extended by properties we think make the configuration easier or are outright lacking:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Extended with</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITestStep</td>
<td>StepNumber(int) – The number of the step as shown in TFS test case work item. Shared steps are automatically marked with a different numbering. A normal step is an integer x, whereas a shared step is marked as x.0 and substeps are inline shaped with numbering x.1, x.2,...</td>
</tr>
<tr>
<td>ITestStepResult</td>
<td>StepNumber(int) – The number of the test step for this result as shown in TFS test case work item. Shared steps are inlined, therefore this value may be different for the same shared step in different test cases. Title(string) – The title of the tested test step. Parameters are replaced with the values of the tested iteration. ExpectedResult(string) – The expected result of the tested test step. Parameters are replaced with the values of the tested iteration. TestStep (TestStep-Object) – The tested test step.</td>
</tr>
<tr>
<td>ISharedStep</td>
<td>Not available. Shared steps are automatically marked.</td>
</tr>
<tr>
<td>ISharedStepResult</td>
<td>Not available. Shared step results are automatically marked.</td>
</tr>
<tr>
<td>ITestAttachment</td>
<td>LocalPath (string) – Path to the local file if the attachment was downloaded. Only available in combination with the AttachmentLink-element LocalFilename (string) – Filename of the local file if the attachment was downloaded. Only available in combination with the AttachmentLink-element</td>
</tr>
<tr>
<td>ITestCase</td>
<td>TestParametersWithAllValues – Enables to display all values for a test parameter in a list. This can be combined with the iteration number.</td>
</tr>
</tbody>
</table>

Hyperlink configuration
In this test reporting extension is implemented support for hyperlinks:

- Common hyperlink
- Hyperlink to viewer of build on Web Access portal of project.
- Hyperlink to editor of work item on Web Access portal of project
- Hyperlink to viewer of work item on Web Access portal of project
- Hyperlink to viewer of work item in specific revision on Web Access portal of project
- Hyperlink to view of test attachments

Therefore is an extension of replacement configuration defined.

Common hyperlink

```xml
<Replacement Bookmark="AttachmentName" Property="Name">
  <Url Link="Uri"/>
</Replacement>
```

As Uri must be defined the full Uri. For example an Uri of attachment.

Hyperlink for viewer of build
As BuildNumber must be defined the build number.

Hyperlink for editor of work item

As Uri must be defined the Team Foundation Server artifact uri.

Hyperlink to viewer of work item

As Id must be defined the id of work item.

Hyperlink to viewer of work item in specific revision

As Id must be defined the id of work item and as Revision the revision of work item.

Formatting of the Hyperlink and generic field mapping

The Format attribute can be used to perform a custom formatting for the linked work item using its fields in the previously mentioned style. The curly brackets are used for identifying the fields and are replaced by the field values.

Remarks:

- When using this the work item itself must be reloaded on the fly. Thus performance of large reporting generation might decrease.
- For HTML fields (e.g. System.Description) a mapping must exist HTML for the specific work item type and the field mapping must be of type HTML or otherwise html content might be displayed.
Hyperlink to viewer of test attachments

The occurrence of an attachment link will trigger downloading the attachment to a subfolder of the document for the modes “DownloadOnly” and “DownloadAndLinkToLocalFile”. The properties “LocalPath” and “LocalFilename” are only available using these modes.

Using the mode “DownloadOnly” in combination with a property other than “LocalPath” or “LocalFilename” is not recommended as the attachments will be downloaded but there is usually no way to tell which a attachments belongs to which occurrence in the document.

In case of using the mode “DownloadAndLinkToLocalFile” there are three options. The attachment folder can be named with a guid, without a guid or based on test suite. This has to be specified in the MappingConfigurations of the w2t file using the property AttachmentFolderMode.

Possible values are WithGuid, WithoutGuid and BasedOnTestSuite, whereas default is the first one.

- **WithGuid** – Name of attachment folder contains GUID. Name of attachment file contains GUID.
- **WithoutGuid** - Name of attachment folder doesn’t contain GUID. Name of attachment file contains GUID.
- **BasedOnTestSuite** – For every test suite exists folder with name and ID of test suite. Inside that test suite folder can be a new folder which represents child test suite or attachment file which name contains name and ID of test case. So, there exists test suite hierarchy with n-depth.

File and folder structure is the following:

- File XYZ.docx (Test Report)
- Folder “TS A (%Test Suite ID)”
  - File “TC A.1 (%Test Case ID%)_0.png”
  - File “TC A.1 (%Test Case ID%)_1.png”
  - File “TC A.1 (%Test Case ID%)_2.jpeg”
- Folder “TS C (%Test Suite ID)”
  - Folder “TS C_II (%Test Suite ID)”
    - File “TC C_II.1 (%Test Case ID)_0.png”
Linked templates
As previously mentioned, in the configuration can be defined the linked template. This means that
bookmark is not replaced with simple text (evaluated property), but with other template.

The configuration telling that the linked template is to use, should be defined as follows:

```xml
<Replacement
    Bookmark="ActionList"
    Property="Actions"
    LinkedTemplate="ActionSimpleList"/>
```

And the definition of template can be defined as follows:

```xml
<Template
    Name="ActionSimpleList"
    FileName="...">
    <Replacements>
        <Replacement
            Bookmark="Title"
            Property="Title.ToPlainText"/>
        <Replacement
            Bookmark="ExpectedResult"
            Property="ExpectedResult.ToPlainText"/>
    </Replacements>
</Template>
```

Querying WorkItems that are linked to TestResults
With version WordToTFS 4.3 a new parameterized property was introduced that enables to
possibility to get all linked WorkItems of a specific Type and LinkType for a test results.

The new property `LinkedWorkItemsForTestResult` executes a query during the import of the
TestResults. The property must be used with the new `Parameter` option, that passes parameters to
the property. Two parameters can be passed to the property, where the first describes the type of
one or multiple work item types. The second parameter describes the link type that links the work
items to the test result.

The property returns a list of work items that fit to the parameters provided.

```xml
<Replacement
    Bookmark="AffectedDefects"
    Property="LinkedWorkItemsForTestResult"
    Parameters="{Defect,Task}{Microsoft.VSTS.Common.TestedBy-Forward}"
    LinkedTemplate="LinkedDefectsTemplate"/>
```

By using a LinkedTemplate the properties of the work items can be accessed as usual.

Nevertheless, this custom property has some restrictions in its use

- As this feature executes another query, the performance during the creation of a report will be slower.
- It is only available for test results in the `TestResultElementTemplate`. Other work items have the build-in property `Links` that can be used to obtain linked elements
- The name of the LinkType must contain the reference name as well as the direction (Forward or Reverse)
Using Object Queries to query additional Data for Test Management Objects

From version 4.4 on it is possible to query additional items that are linked to objects in the Test Management of TFS.

\[\text{ObjectQueries}\]

\[\text{ObjectQuery Name=LinkedDefectsForTestSuite}\]

\[\text{DestinationElements}\]

\[\text{DestinationElement WorkItemType =Defect}\]

\[\text{WorkItemLinkFilters FilterType =Include}\]

\[\text{Filter LinkType=Microsoft.VSTS.Common.TestedBy-Forward} /\]

\[\text{Filter LinkType=Microsoft.VSTS.Common.TestedBy-Reverse} /\]

\[\text{Filter LinkType=System.LinkTypes.Related-Forward} /\]

\[\text{Filter LinkType=Microsoft.VSTS.Common.Affects-Forward} /\]

\[\text{Options Distinct=true}\]

\[\text{ObjectQuery}\]

\[\text{ObjectQueries }\]

Definition of object queries

Within the ObjectQueries element you can define multiple ObjectQuery Elements. Each element defines a query that is executed during the creation of a report. The result of this queries are always a list of objects, which can be used during a replacement of properties.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The unique name of the object query.</td>
</tr>
</tbody>
</table>

Destination Elements

The <DestinationElements> defines what types of elements will be contained in the returning list of the object queries. You can search for any work item type. E.g. “Bug” or “Change Request”. In addition it is also possible to query for elements that are no work items at the moment.

<table>
<thead>
<tr>
<th>Value of attribute “WorkItemType”</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Builds</td>
<td>Find all used builds in a TestPlan, a hierarchy of TestSuites or a TestCase</td>
</tr>
<tr>
<td>Configurations</td>
<td>Find all used Configurations for a TestPlan, a hierarchy of TestSuites or a TestCase</td>
</tr>
<tr>
<td>Test Results</td>
<td>Find all connected Test Results for a TestSuite, a TestPlan or a TestCase</td>
</tr>
<tr>
<td>Test Cases</td>
<td>Find all connected Test Cases for a TestSuite or a TestPlan</td>
</tr>
<tr>
<td>Test Points</td>
<td>Find all connected Test Points for a TestSuite or a TestPlan. The Test Points serve as joint between Test Cases and Test Results. More information can be found in the ITestPoint interface.</td>
</tr>
</tbody>
</table>

WorkItemLinkFilters (optional)

In addition you can specify a filter that will filter the returning work items by the way the work items are linked.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterType</td>
<td>Specifies the type of the filter. Possible values are “Include” or “Exclude”. If no filter is specified, an include filter will be used.</td>
</tr>
</tbody>
</table>
### Options
In addition to the WorkItemType and the LinkType it is also possible to provide various options that will alter the returning list.

<table>
<thead>
<tr>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinct</td>
<td>The immutable name of the link that is used in TFS</td>
</tr>
<tr>
<td>FilterProperty</td>
<td>By specifying a FilterProperty the distinct options can be further customized. You can filter by any property of the corresponding object</td>
</tr>
<tr>
<td>Latest (Only for TestResults)</td>
<td>This option will make sure, that the returning list will only contain the latest TestResults</td>
</tr>
</tbody>
</table>

### Usage of Object Queries
The result of an object query can be used in combination with a linked template (See: Hyperlink to viewer of test attachments)

The occurrence of an attachment link will trigger downloading the attachment to a subfolder of the document for the modes “DownloadOnly” and “DownloadAndLinkToLocalFile”. The properties “LocalPath” and “LocalFilename” are only available using these modes.

Using the mode “DownloadOnly” in combination with a property other than “LocalPath” or “LocalFilename” is not recommended as the attachments will be downloaded but there is usually no way to tell which attachments belong to which occurrence in the document.

In case of using the mode “DownloadAndLinkToLocalFile” there are three options. The attachment folder can be named with a guid, without a guid or based on test suite. This has to be specified in the `MappingConfigurations` of the w2t file using the property `AttachmentFolderMode`.

```xml
<MappingConfiguration xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                        xsi:noNamespaceSchemaLocation="W2T.xsd"
                        ShowName="CMMI(2015)"
                        AttachmentFolderMode="BasedOnTestSuite">
    <Template Bookmark="AttachmentSimpleList" Property="...">
        <Replacements>
            <Replacement Bookmark="Name" Property="Name">
                <AttachmentLink Mode="DownloadAndLinkToLocalFile"/>
            </Replacement>
        </Replacements>
    </Template>
    ...
</MappingConfiguration>
```
Possible values are *WithGuid*, *WithoutGuid* and *BasedOnTestSuite*, whereas default is the first one.

- **WithGuid** – Name of attachment folder contains GUID. Name of attachment file contains GUID.
- **WithoutGuid** - Name of attachment folder doesn’t contain GUID. Name of attachment file contains GUID.
- **BasedOnTestSuite** – For every test suite exists folder with name and ID of test suite. Inside that test suite folder can be a new folder which represents child test suite or attachment file which name contains name and ID of test case. So, there exists test suite hierarchy with n-depth.

File and folder structure is the following:

- File XYZ.docx (Test Report)
- Folder “TS A (%Test Suite ID)”
  - File “TC A.1 (%Test Case ID%)_0.png”
  - File “TC A.1 (%Test Case ID%)_1.png”
  - File “TC A.1 (%Test Case ID%)_2.jpeg”
- Folder “TS C (%Test Suite ID)”
  - Folder “TS C_II (%Test Suite ID)”
    - File “TC C_II.1 (%Test Case ID)_0.png”

Linked templates ). The Name of the property should correspond to the name of your defined object query

```xml
<Replacement Bookmark="AllLinkedDefectsForTestSuite"
Property="LinkedDefectsForTestSuite" LinkedTemplate="ShowResultsDefects" />
```

### Console Extension

Since version 4.5 the additional file *WordToTFS.exe* is shipped out with the installation of WordToTFS. This extension allows the use of all WordToTFS features without the need of any user interaction. It is therefore possible to create documents that contain work items, test result reports and test specification reports using the command line provided by windows. Possible use cases are:

- The automatic creation of a report that shows the latest test results
- An on-demand creation of a document that contains the latest state of a requirement

### Installing the Console Extension

WordToTFS.exe is shipped out automatically with the installation of WordToTFS. It is therefore only necessary to execute the file. This can be achieved by either:

1. Calling WordToTFS.exe directly from the installation path
2. Adding the location of the WordToTFS.exe to the PATH environment variable of the user and executing it from any path

The Settings window from WordToTFS offers the functionality to determine the installation path or to add the executable to the PATH environment variable. If you want to access the exe directly you can use the Open install path button (1) to go to the installation path directly.

By activating the console extension (2) the installation path is added to the PATH environment variable of the current user. This will make the exe accessible by typing `wordtotfs` in the command line.

Please Note: A restart is recommended to ensure that all environment variables are correctly loaded.

---

**Commands**

The WordToTFS Console Extension provides the same basic functionality as WordToTFS itself. Therefore four basic commands can be used to create reports, which are described in this chapter.

The basic settings for all reports like the server name or the name of the team project as well as arguments for the reports can be specified in a configuration file. The configuration file has to be referenced in each command by the option “co” or “ConfigFile”. For more information see Chapter Customization of the Configuration File.

**GetWorkItems-Command**

The GetWorkItems-Command creates a word document containing the specified work items. It has two options, which allows to get work items on two different ways. The options are exclusive, that is to say they can’t be used in combination.

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetWorkItems</code></td>
<td></td>
</tr>
<tr>
<td><code>GetWorkItems --server</code></td>
<td></td>
</tr>
<tr>
<td><code>GetWorkItems --team</code></td>
<td></td>
</tr>
<tr>
<td><code>-workitem</code></td>
<td></td>
</tr>
<tr>
<td><code>-workitemnum</code></td>
<td></td>
</tr>
<tr>
<td><code>-workitemids</code></td>
<td></td>
</tr>
</tbody>
</table>
Further the GetWorkItems-Command has options to configure the basic settings without a configuration file. Therefore the following options are provided.

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>w or WorkitemIDs</td>
<td>Work items can be specified by IDs separated by commas.</td>
</tr>
<tr>
<td>q or WorkitemQuery</td>
<td>Work items can be specified by a predefined query. The query has to be given in the following form: “NameOfTheTeamProject/QueryFolder/QueryName”</td>
</tr>
</tbody>
</table>

Example Calls:

1. GetWorkItems -w = ”49,723” -s = ”devtfs2013\WordToTFSDev”  
   -p = “CMMIProject” -t = “MSF for CMMI (2013)”  
   -f = “OutputFileName” -h = “true” -o = “true” -l = “0”  

2. GetWorkItems -q = ”MyTeamProject/My Shared Queries/My Saved Query” -co = “ConfigurationFilePath”

TestSpecReport-Command

The TestSpecReport-Command creates a test specification report and has no options. All necessary information are configured in the configuration file.

Example Call:

TestSpecReport -co = “ConfigurationFilePath”
TestSpecReportByQuery
The TestSpecReport-Command creates a test specification report by query. The same two options as in the GetWorkItems-Command are provided, which can’t be used in combination.

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>w or WorkItemIDs</td>
<td>Work items can be specified by IDs separated by commata.</td>
</tr>
<tr>
<td>q or WorkItemQuery</td>
<td>Work items can be specified by a predefined query. The query has to be given in the following form: “NameOfTheTeamProject/QueryFolder/QueryName”</td>
</tr>
</tbody>
</table>

Example Calls:
1. TestSpecReportByQuery -w = "52,68,73" -co = “ConfigurationFilePath”
2. TestSpecReportByQuery -q = “MyTeamProject/My Shared Queries/My Saved Query” -co = “ConfigurationFilePath”

TestResultReport-Command
The TestResultReport-Command creates a test result report and has no options. All necessary information are configured in the configuration file.

Example Call:
TestSpecReport -co = “ConfigurationFilePath”

Customization of the Configuration File
The configuration file is an XML-file used to set the basic settings as well as the arguments for the test reports. Therefore it has the two parts Settings and TestConfiguration, whereas only the Settings-Part is mandatory. The TestConfiguration-Element does not have to be specified if only the GetWorkItems-Command is used.

You can download additional configuration samples including some sample calls from: http://wordtotfs.aitag.com/2013/Download/ConfigurationSamples.zip

Example:
```xml
<?xml version="1.0"?>
<Configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Settings ... />
    <TestConfiguration>...
    </TestConfiguration>
</Configuration>
```
Basic Settings
The basic settings are defined within the Settings(Xml)-Element. It includes the elements shown in the following table.

<table>
<thead>
<tr>
<th>Element</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Name of the server in the form: “ServerName/CollectionName”</td>
</tr>
<tr>
<td>Project</td>
<td>Name of the team project.</td>
</tr>
<tr>
<td>Template</td>
<td>Name of the WordToTFS template, which should be used for the generation of the report.</td>
</tr>
<tr>
<td>Filename</td>
<td>Path where the generated document should be stored.</td>
</tr>
<tr>
<td>WordHidden</td>
<td>Determines whether the word document should be opened during report generation. Default value is false, that is to say that the document is opened.</td>
</tr>
<tr>
<td>Overwrite</td>
<td>Determines whether the specified file should be overwritten if it already exists. Default value is false.</td>
</tr>
<tr>
<td>Close</td>
<td>Determines whether the word document should be closed after report generation (only necessary of –h=false). Default value is false.</td>
</tr>
<tr>
<td>DebugLevel</td>
<td>Version of the debug level, which should be used for logging messages. Default value is 0. Available values:</td>
</tr>
<tr>
<td></td>
<td>• 0 – Off</td>
</tr>
<tr>
<td></td>
<td>• 1 – Error</td>
</tr>
<tr>
<td></td>
<td>• 2 – Warning</td>
</tr>
<tr>
<td></td>
<td>• 3 – Info</td>
</tr>
<tr>
<td></td>
<td>• 4 – Verbose</td>
</tr>
</tbody>
</table>

Example:

```xml
<?xml version="1.0"?>
<Configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Settings>
    Server="devtfs2013\WordToTFSDev"
    Project="CMMIProject"
    Template="MSF for CMMI (2013)"
    Filename="C:\YourPath\%TIMESTAMP%\FileName.dotx"
    WordHidden="false"
    Overwrite="true"
    Close="true"
    DebugLevel="0"/>
  ...
</Configuration>
```

Test Report Settings
The settings for the test reports are defined within the TestConfiguration-Xml-Element. It can have two parts – the TestSpecificationConfiguration-part and the TestResultConfiguration-part.

The TestSpecificationConfiguration-part has the following elements:
Element | Explanation
--- | ---
TestPlan | Name of the test plan, for which the report should be generated.
TestSuite | Selected test suite in the test plan. If no test suite is specified, the root test suite is taken.
CreateDocumentStructure | Determines whether document structure should be created.
DocumentStructure | If CreateDocumentStructure is set to true, one of the following options has to be specified:
  • AreaPath
  • IterationPath
  • TestPlanHierarchy
SkipLevels | Determines the skip levels (cf. WordToTFS User Guide). Has to be an integer.
IncludeTestConfigurations | Determines whether test configurations are included in the test specification report.
TestConfigurationsPosition | If IncludeTestConfigurations is set to true, one of the following options has to be specified:
  • AboveTestPlan
  • BeneathFirstTestSuite
  • BeneathTestPlan
  • BeneathTestSuites
SortTestCasesBy | Determines how the test cases are sorted. The following options are available:
  • AreaPath
  • IterationPath
  • WorkItemId
  • No Sorting

Example:

```xml
<TestConfiguration>
  <TestSpecificationConfiguration>
    TestPlan="MyPlan"
    TestSuite="MySuite"
    CreateDocumentStructure="true"
    DocumentStructure="TestPlanHierarchy"
    SkipLevels="2"
    IncludeTestConfigurations="true"
    TestConfigurationsPosition="AboveTestPlan"
    SortTestCasesBy="WorkItemId"
  </TestSpecificationConfiguration>
</TestConfiguration>
```

The TestResultConfiguration part has the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>A specific build can be specified. Default is “All”</td>
</tr>
</tbody>
</table>
| TestConfiguration | Determines for which test configuration the results are shown. Default is “All”.
<p>| TestPlan        | Name of the test plan, for which the report should be generated. |
| TestSuite       | Selected test suite in the test plan. If no test suite is specified, the root test suite is taken. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateDocumentStructure</td>
<td>Determines whether document structure should be created.</td>
</tr>
</tbody>
</table>
| DocumentStructure               | If CreateDocumentStructure is set to true, one of the following options has to be specified:  
- AreaPath  
- IterationPath  
- TestPlanHierarchy |
| SkipLevels                      | Determines the skip levels (cf. WordToTFS User Guide). Has to be an integer.  |
| IncludeTestConfigurations       | Determines whether test configurations are included in the test specification report. |
| TestConfigurationsPosition      | If IncludeTestConfigurations is set to true, one of the following options has to be specified:  
- AboveTestPlan  
- BeneathFirstTestSuite  
- BeneathTestPlan  
- BeneathTestSuites |
| SortTestCasesBy                 | Determines how the test cases are sorted. The following options are available:  
- AreaPath  
- IterationPath  
- WorkItemId  
- No Sorting |
| IncludeOnlyMostRecentResults    | Determines whether only the most recent test results are included in the report or not. |
| MostRecentForAllConfigurations  | If IncludeOnlyMostRecentTestResults is set to true, it determines if the most recent results should be included for all configurations. |

**Example:**

```xml
<TestConfiguration>
  <TestResultConfiguration
    Build="All"
    TestConfiguration="Windows 8"
    TestPlan="MyPlan"
    TestSuite="MySuite"
    CreateDocumentStructure="true"
    DocumentStructure="TestPlanHierarchy"
    SkipLevels="2"
    IncludeTestConfigurations="true"
    TestConfigurationsPosition="AboveTestPlan"
    SortTestCasesBy="WorkItemId"
    IncludeOnlyMostRecentResults="true"
    MostRecentForAllConfigurations="true" />
</TestConfiguration>
```
FAQ

Q Installation fails with “The following Microsoft Office Solution cannot be loaded because a compatible version of the .NET Framework is not installed.”
A Make sure you have both .NET 3.5 SP1 and .NET 4.0 installed

Q Why do I get “Revision conflicts” when trying to publish although I have refreshed to the most recent version and I am sure nothing has changed on the TFS?
A You may have forgotten to map the System.Rev field which is necessary for this check to work correctly. Also you may have selected a template that uses a “PublishOnly” direction which means your work item will never be refreshed to the latest version.

Q Why is text formatting sometimes not published to the TFS when using HTML fields?
A Word exports some formatting only when it considers the item finished. This is not automatically the case for the last formatting item within a cell. In this case, you have to make sure that the last symbol in an end-of-paragraph marker (by pressing Enter).

Q Import / Export of items does not work for a specific work item but it does for others
A Make sure, the table layout of the affected work item does not contain vertically merged cells. Cell addressing by row and column does not work with vertically merged cells.

Q Template is not shown in Word
A In case a custom template is not shown in the list of available Template, verify that the template location is added using the template manager and that the w2t is correct. In case of syntax or logical errors, the template will not be shown as available template in Word.

Q I am using Visual Studio Team Services (VSTS). Some picture (e.g. in the description field of a work item) are not shown.
A In case you are using VSTS some content may be blocked by your security settings. Add https://*.visualstudio.com to the trusted sites in your internet settings.

Q I am creating big reports with the test reporting functionality of WordToTFS. After some time the creation of the report fails with an exception that states “Command Failed” or an access exception.
A The test reporting functionality of WordToTFS created temporary word documents in your temp folder. If your antivirus software is scanning this folder, it can block the access to these file with a read/write lock. This will raise cause that an exception is raised by Word. To overcome this issue exclude the temp path from scanning. The exact path should be stated in the raised exception. Depending on your configuration the temp folder of WordToTFS can have various locations
- The path specified by the TMP environment variable.
- The path specified by the TEMP environment variable.
- The path specified by the USERPROFILE environment variable.