

TFS Practical Reporting Guide - Part 2

Data Warehouse

Visual Studio ALM Rangers

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft grants you a license to this document under the terms of the Creative Commons Attribution 3.0 License. All other rights are reserved.

© 2013 Microsoft Corporation.

Microsoft, Active Directory, Excel, Internet Explorer, SQL Server, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

Foreword	4
Introduction	5
Data Warehouse Secrets.....	6
Reporting Documentation and Tooling Gems	6
CodePlex Gems.....	7
TFS 2013 Charting.....	8
Practical Reports.....	9
Build Duration Report	9
Build Duration Over Time Report.....	14
Build Status Over Time Report	15
Mean Time Between Failure (MTBF) Report	17
Mean Time To Resolution (MTTR) Report	19
Time Tracking Report	21
Custom Report Service.....	25
Data Adapters and the TFS Warehouse	28
TFS Warehouse Adapter Basics.....	29
In Conclusion	31
Appendix	32
A report template to start with	32
Customizing your VM for Analysis Services.....	35
Understanding the Work Item Store Relational Data Model	36

Foreword

One of the key value propositions for Team Foundation Server has been the unified view of reporting data crossing the complete Application Lifecycle Management (ALM) spectrum. In the box, we've created a broad selection of reports that leverage this data to help give insights into project status. Obviously, it's not possible for the product group to build every report or anticipate how the data can be used by every team. The ALM Rangers have done a great job extending our existing reporting surface area by bringing additional insights into build health, team focus and reaction to customers. These reports can serve as tools for your team to improve efficiencies and become more predictable at delivering value to your customers. As the product group owner for reporting, I'm always looking for new, insightful ways to make teams more productive and we'd love feedback on how these reports have improved your teams and what additional ideas you have on reports that we may want to author in the future.

Justin Marks – Senior Program Manager, Cloud Dev Services

Introduction

Welcome to TFS Reporting Guide where we, the ALM Rangers, will take you on a fascinating journey to discover the exciting and powerful reporting features provided by Team Foundation Server.

This guide focuses on providing practical guidance and samples to enable Team Foundation Server users to create valuable reports using the TFS Data Warehouse, based on real-world scenarios.

Intended audience

We expect the majority of our audience to be developers; however, the guide is for everyone from the developer to the administrator to create and use reports that help ensure that Team Foundation Server and associated projects are well tuned and in a healthy state. The guide assumes a basic knowledge of the Team Foundation Server and a reporting mindset – in other words, intermediate to advanced Team Foundation Server users.

What you'll need

The following prerequisites needed and referenced in this guide as *supported editions*:

- Visual Studio 2012, 2013
- Team Foundation Server 2010, 2012, 2013
- Microsoft Office Excel
- SQL Server Business Intelligence Development Studio

Visual Studio ALM Rangers

The Visual Studio ALM Rangers are a special group with members from the Visual Studio Product group, Microsoft Services, Microsoft Most Valuable Professionals (MVP) and Visual Studio Community Leads. Their mission is to provide out-of-band solutions to missing features and guidance. A growing Rangers Index is available [online](#)¹.

Contributors

[Grant Holliday](#)², [Hamid Shahid](#)³, [Jesse Houwing](#)⁴, [Jim Szubryt](#)⁵, [Mattias Sköld](#)⁶, [Prasanna Ramkumar](#)⁷, [Priyanka Janardhan](#)⁸

Additional ALM Rangers Resources

Understanding the ALM Rangers – <http://aka.ms/vsarunderstand>

Visual Studio ALM Ranger Solutions – <http://aka.ms/vsarsolutions>

Using the sample source code, Errata and support

All source code in this guide is available for download via the [TFS Practical Reporting](#)⁹ home page where we also provide the latest corrections and updates. Download the file **eBook2-Package.exe**.

¹ <http://aka.ms/vsarindex>

² <http://blogs.msdn.com/b/granth/>

³ http://blogs.msdn.com/b/willy-peter_schaub/archive/2013/03/25/introducing-the-visual-studio-alm-rangers-hamid-shahid.aspx

⁴ http://blogs.msdn.com/b/willy-peter_schaub/archive/2012/07/03/introducing-the-visual-studio-alm-rangers-jesse-houwing.aspx

⁵ http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/07/31/introducing-the-visual-studio-alm-rangers-jim-szubryt.aspx

⁶ http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/03/28/introducing-the-visual-studio-alm-rangers-mattias-sk-246-ld.aspx

⁷ http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/11/23/introducing-the-visual-studio-alm-rangers-prasanna-ramkumar.aspx

⁸ http://blogs.msdn.com/b/willy-peter_schaub/archive/2013/04/03/introducing-the-visual-studio-alm-rangers-priyanka-janardhan.aspx

⁹ <http://aka.ms/treasure55>

Data Warehouse Secrets

If you are new to TFS Administration, you may not have heard of [Grant Holliday's](#) ¹⁰ excellent reports he had developed for Team Foundation Server. Grant is a Senior Service Engineer for the VS Online and spends a majority of his time in the field working with companies on their TFS implementations.

Grant had developed the first Report Pack back in March of 2009 and has kept it aligned with each TFS version. There are two sets of administrative reports: the [Performance Pack](#) ¹¹ and the [Administrative Pack](#) ¹². The Performance pack consists of five reports:

- Execution Time Summary
- Server Status – Source Control Request Queue
- Server Status – Top Users Bypassing Proxies
- Server Status – Historical Performance Trends
- Server Status – Recent Performance Trends

The Administrative pack consists of three reports:

- Cube Status
- Blocked Fields
- Reportable Fields

With all of these reports, you need to upload them to your Reporting Server Services instance, update the security information for the datasets and verify that the connection can be made.

NOTE

You might need to modify the queries in the report files to remove the database and schema names from the table to be queried. For example, a query with the clause "FROM TfsActivityLogging.dbo.tbl_Command with (nolock)" will become "FROM tbl_Command with (nolock)".

With TFS 2012 (and 2013) there is a new feature accessible by TFS administrators named Activity Log. This feature can be found at http://your-server:8080/tfs/_oi. Grant posted [Information](#) ¹³ about how to view this information. The data displayed in these pages is straight from the TFS operational (transactional) tables and adheres to Microsoft's best practices on access production data. The screens associated with this information will continue to evolve over time and you should expect to see more detailed information in our Reporting guidance.

Reporting Documentation and Tooling Gems

The following table summarizes other content on reporting that you should peruse and reference:

Gem	Description
Administrative Report Pack for Team Foundation Server 2010 ¹⁴ Announcing TFS Performance Report Pack ¹⁵ TFS2010: SQL Queries for TFS Statistics ¹⁶	Reports (based on SQL Server Reporting Services) that can be used to evaluate and get a picture of the status of some of the internals of your TFS environment

¹⁰ <http://blogs.msdn.com/b/granth>

¹¹ <http://blogs.msdn.com/b/granth/archive/2009/02/03/announcing-tfs-performance-report-pack.aspx>

¹² <http://blogs.msdn.com/b/granth/archive/2010/07/12/administrative-report-pack-for-team-foundation-server-2010.aspx>

¹³ <http://blogs.msdn.com/b/granth/archive/2013/02/13/tfs2012-new-tools-for-tfs-administrators.aspx>

¹⁴ <http://blogs.msdn.com/b/granth/archive/2010/07/12/administrative-report-pack-for-team-foundation-server-2010.aspx>

¹⁵ <http://blogs.msdn.com/b/granth/archive/2009/02/03/announcing-tfs-performance-report-pack.aspx>

¹⁶ <http://blogs.msdn.com/b/granth/archive/2009/10/23/tfs2010-sql-queries-for-tfs-statistics.aspx>

Gem	Description
TFS Internal Usage Statistics - 1st Half CY 2013 ¹⁷ ¹⁸ Warehouse and Job Service Administrator Reports ¹⁹	
Data Driven Subscription Reporting a la Grant ²⁰ Monitoring the TFS Data Warehouse - FAQ ²¹ How to generate a report of active users who log onto the TFS server ²²	Blog posts
Professional Application Lifecycle Management ²³	Book covering reporting, portals and dashboards, written by Microsoft insiders and MVPs.
Professional Team Foundation Server ²⁴	Book covering reporting and SharePoint Dashboards, written by Microsoft insiders and MVPs.
TFS Planning and DR Avoidance Guide ²⁵	Discussions around reporting in terms of disaster recovery avoidance and monitoring.

Table 1 – Reporting Documentation and Tooling Gems

CodePlex Gems

We would be remiss not to highlight the great work done by Microsoft ALM Rangers to contribute back to the community through CodePlex projects.

Gem	Description
Community TFS Report Extensions ²⁶	A Community project that provides extended reporting for Team Foundation Server installations. The intent of this project is to allow for a single repository of SQL Server Reporting Services report for Team Foundation 2010 and above.
GoDev for TFS ²⁷	GovDev is an open source, TFS Process Template that combines the formality of CMMI/Waterfall with the flexibility of Agile/Iterative.
TFS Scorecard ²⁸	TFS Scorecard provides insight into a TFS environment and an organization's TFS adaptation. You can see trends, activities, source control, work item tracking as well as the overall adaptation.

Table 2 – Reporting projects on CodePlex

¹⁷ <http://blogs.msdn.com/b/visualstudioalm/archive/2013/08/21/10443052.aspx>

¹⁸ Take special note of the update for TFS2012 in 2nd comment of the blog post

¹⁹ <http://blogs.msdn.com/b/granth/archive/2010/02/07/tfs2010-warehouse-and-job-status-reports.aspx>

²⁰ http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/01/31/tfs-integration-platform-data-driven-subscription-reporting-a-la-grant.aspx

²¹ <http://blogs.msdn.com/b/granth/archive/2010/07/12/monitoring-the-tfs-data-warehouse-faq.aspx>

²² <http://blogs.msdn.com/b/tfssetup/archive/2013/09/18/how-to-generate-a-report-of-active-users-who-log-onto-the-tfs-server.aspx>

²³ <http://aka.ms/treasure10>

²⁴ <http://aka.ms/treasure16>

²⁵ <http://aka.ms/treasure5>

²⁶ <https://communitytfsreports.codeplex.com>

²⁷ <http://govdevfortfs.codeplex.com>

²⁸ <https://communitytfsreports.codeplex.com>

TFS 2013 Charting

As this guide is being published, TFS 2013 has just been released to manufacturing. The Microsoft ALM Rangers will release future versions of the reporting guide to support the new features in the product.

One of the exciting features released at RTM is [Work Item Charting](#)²⁹. Incorporating reports into the TFS web experience will be coming in future versions of TFS. Brian Harry blogged about the plans which include:

- trend charts
- pinning charts to the home page
- aggregate functions
- supporting more data than just work items

Microsoft will be improving quickly in the reporting space. We should expect to see new features frequently on the VS Online and it make its way to on-premises TFS with the quarterly updates.

Here's an example of what you can create with Work Item Charting.

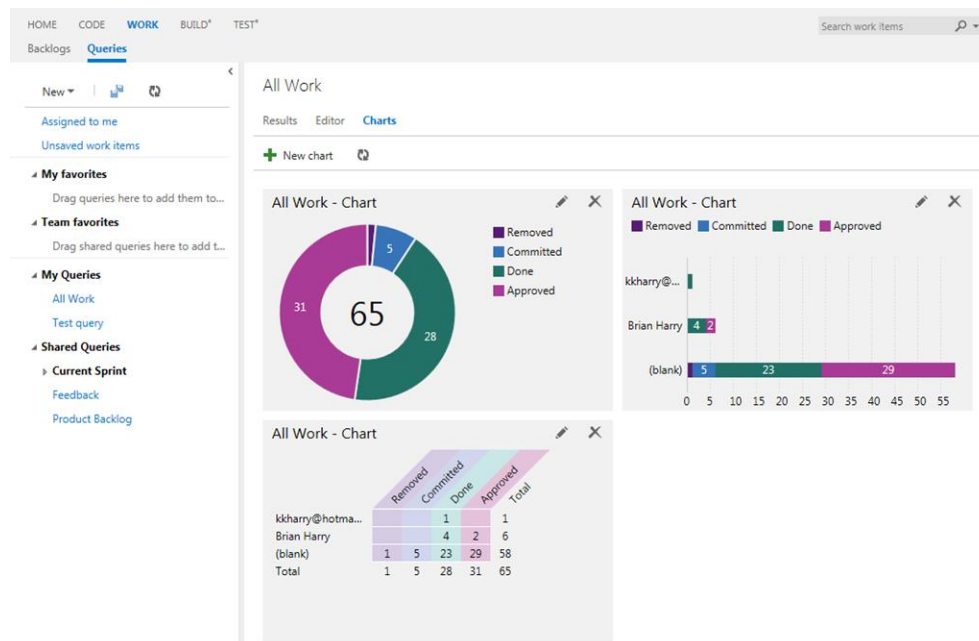


Figure 1 - TFS 2013 Charting

²⁹ <http://blogs.msdn.com/b/bharry/archive/2013/09/09/team-foundation-service-update-sept-9.aspx>

Practical Reports

Build Duration Report

Context	<p>If you want to optimize your team build and find out which part of the build takes the most time, Visual Studio's Build log is sometimes difficult to use. Visual Studio does not make it easier by making you scroll through endless series of logs.</p> <p>This report provides a useful alternative by showing you the details of each step and duration in a non-hierarchical way.</p>
Version	TFS 2010, TFS 2012
Category	Health reporting

Table 3 – Build duration report

Home > Custom Reports > Build Duration Report									
<div> <div>1 of 1</div> <div>100%</div> <div>Find Next</div> </div>									
Build Duration Report									
Node Id	Parent Id	Type Name	Last Modified Date	Display Text	Start Time	Finish Time	Duration Seconds	Duration Minutes	Server Path
1		ActivityTracking	10/11/2013 10:32:11 PM	Overall Build Process	10/11/2013 10:31:44 PM	10/11/2013 10:32:11 PM	27	0.45	
2	1	ActivityTracking	10/11/2013 10:31:44 PM	Update Build Number	10/11/2013 10:31:44 PM	10/11/2013 10:31:44 PM	0	0.00	
3	1	ActivityTracking	10/11/2013 10:31:44 PM	Create the Drop Location	10/11/2013 10:31:44 PM	10/11/2013 10:31:44 PM	0	0.00	
4	1	AgentScopeActivityTracking	10/11/2013 10:32:11 PM	Run On Agent	10/11/2013 10:31:44 PM	10/11/2013 10:32:11 PM	27	0.45	
5	4	ActivityTracking	10/11/2013 10:31:59 PM	Delete Test Results Directory	10/11/2013 10:31:59 PM	10/11/2013 10:31:59 PM	0	0.00	
6	4	ActivityTracking	10/11/2013 10:31:59 PM	Delete Binaries Directory	10/11/2013 10:31:59 PM	10/11/2013 10:31:59 PM	0	0.00	
7	4	ActivityTracking	10/11/2013 10:32:00 PM	Delete Workspace	10/11/2013 10:31:59 PM	10/11/2013 10:32:00 PM	1	0.02	
8	4	ActivityTracking	10/11/2013 10:32:00 PM	Delete Sources Directory	10/11/2013 10:32:00 PM	10/11/2013 10:32:00 PM	0	0.00	
9	4	ActivityTracking	10/11/2013 10:32:00 PM	Create Workspace	10/11/2013 10:32:00 PM	10/11/2013 10:32:00 PM	0	0.00	
10	4	ActivityTracking	10/11/2013 10:32:05 PM	Get Workspace	10/11/2013 10:32:00 PM	10/11/2013 10:32:05 PM	5	0.08	
11	4	ActivityTracking	10/11/2013 10:32:05 PM	Create Label	10/11/2013 10:32:05 PM	10/11/2013 10:32:05 PM	0	0.00	
13	4	ActivityTracking	10/11/2013 10:32:09 PM	Compile, Test, and Associate Changesets and Work Items	10/11/2013 10:32:05 PM	10/11/2013 10:32:09 PM	4	0.07	
14	13	ActivityTracking	10/11/2013 10:32:09 PM	Compile and Test	10/11/2013 10:32:05 PM	10/11/2013 10:32:09 PM	4	0.07	
15	14	ActivityTracking	10/11/2013 10:32:09 PM	Run MSBuild for Project	10/11/2013 10:32:05 PM	10/11/2013 10:32:09 PM	4	0.07	
17	13	ActivityTracking	10/11/2013 10:32:09 PM	Associate Changesets and Work Items	10/11/2013 10:32:06 PM	10/11/2013 10:32:06 PM	0	0.00	
18	15	BuildProject	10/11/2013 10:32:09 PM		10/11/2013 10:32:08 PM	10/11/2013 10:32:09 PM	1	0.02	\$/SandboxTfs2010/Source/DeploymentEngine

Figure 2 – Build duration report

Prerequisites

- The user needs access to the TFS_<CollectionName> database, where <CollectionName> is the name of the Team collection.

Sample

TFS 2010

The following code works for TFS 2010. In the example we set the @BuildId to be the latest build. If you wanted the results for a particular build then you would need to supply the build id value.

- **Tbl_build** – The table stores the record for each team build.
- **Tbl_InformationType** – The table stores information of each of the step executed as part of team build.
- **Tbl_buildInformation** – The build information table store details of the result of the execution of each step defined in build information type in each build. The table schema is changed between TFS 2010 and TFS 2012. The TFS 2012 version of the table stores information in a column called fields while the TFS 2010 version stores information in a separate table called tbl_BuildInformationFields.
- **Tbl_buildInformationFields** – The table contains the information fields per row in tbl_BuildInformation for each build. The amount of fields varies per BuildInformationType.

Code 1 – Build duration report TFS 2010

```

DECLARE @BuildId int;
-- Using this SELECT as just an example to get the latest build to view the details of
SELECT @BuildId = MAX(BuildId) FROM tbl_BuildInformation bi WITH(NOLOCK);

SELECT bi.NodeId
,bi.ParentId
,bity.TypeName
,bi.LastModifiedDate
,dpt.FieldValue DisplayText
,CONVERT(DATETIME2, st.FieldValue) StartTime
,CONVERT(DATETIME2, ft.FieldValue) FinishTime
,DATEDIFF(SECOND, st.FieldValue, ft.FieldValue) AS 'Duration (Seconds)'
,CAST((DATEDIFF(SECOND, st.FieldValue, ft.FieldValue))/60.0 AS DECIMAL (12,2)) AS 'Duration (Minutes)'
,sp.FieldValue ServerPath
FROM tbl_BuildInformation bi WITH(NOLOCK)
INNER JOIN tbl_BuildInformationType bity ON bi.NodeType = bity.NodeType
LEFT OUTER JOIN (SELECT NodeId,
    FieldValue
    FROM tbl_BuildInformationField WITH(NOLOCK)
    WHERE FieldName = 'DisplayText') dpt ON bi.NodeId = dpt.NodeId
LEFT OUTER JOIN (SELECT NodeId,
    FieldValue
    FROM tbl_BuildInformationField WITH(NOLOCK)
    WHERE FieldName = 'StartTime') st ON bi.NodeId = st.NodeId
LEFT OUTER JOIN (SELECT NodeId,
    FieldValue
    FROM tbl_BuildInformationField WITH(NOLOCK)
    WHERE FieldName = 'FinishTime') ft ON bi.NodeId = ft.NodeId
LEFT OUTER JOIN (SELECT NodeId,
    FieldValue
    FROM tbl_BuildInformationField WITH(NOLOCK)
    WHERE FieldName = 'ServerPath') sp ON bi.NodeId = sp.NodeId
WHERE BuildId = @BuildId
AND bity.TypeName NOT IN ('BuildMessage',
    'BuildWarning',
    'ExternalLink',
    'ConfigurationSummary')
ORDER BY NodeId;

```

TFS 2012

The database schema has changed between TFS 2010 and TFS 2012 and the table tbl_BuildInformation does not exist anymore. So, the above mentioned query won't work on TFS 2012. An alternate approach is to use the Team Foundation Server's API, use the interface available and get the report via a custom service. In this case, we have created a WCF service with the following interface.

Code 2 – Build duration report TFS 2012 IBuildReports interface

```

/// <summary>
/// The IBuildReports service interface provide methods for a the various build reports
/// created from Team Foundation Server's service
/// </summary>
[ServiceContract]
public interface IBuildReports
{
    /// <summary>
    /// Returns the Build Step Duration report for a given build

```

```

/// </summary>
/// <param name="projectCollectionUri">The Project collection Uri</param>
/// <param name="teamProjectName">The Team Project Name</param>
/// <param name="buildDefinitionName">The build definition name for the build for
    which the report should return the build duration. Only the latest build is
    extracted.</param>
/// <param name="reportOption">An enumeration that determines which steps should be
    returned</param>
/// <returns>A list of build steps with their respective duration</returns>
[OperationContract]
IEnumerable<BuildStep> GetBuildStepDurationReport(string projectCollectionUri,
    string teamProjectName, string buildDefinitionName,
    BuildStepDurationReportOption reportOption);
}

```

The interface has one operation for the Build Step Duration Report. The operation takes in the following parameters and returns an enumeration of Build Steps that contain information about steps such as Name, Start Time, End Time and duration.

Parameter	Details
projectCollectionUri	The URI of the team project collection to connect to.
teamProjectName	The name of the team project that contains the build.
buildDefinitionName	The name of the build definition.
reportOptions	An enumeration with the following possible options: <ul style="list-style-type: none"> All – Returns all the build steps SlowestTen – Returns the ten longest running steps. FastestTen – Returns the ten fastest running steps.

Table 4 – Build duration report TFS 2012 parameters

The return type is an enumeration of objects of type “BuildStep” –which holds information about each build step and its execution time. The Build Step type looks like this:

Code 3 – Build duration report TFS 2012 BuildStep type

```

[DataContract]
public class BuildStep
{
    /// <summary>
    /// Gets or sets the name of the Build Step
    /// </summary>
    [DataMember]
    public string Name { get; set; }

    /// <summary>
    /// Gets or sets the Start Time of the Build Step
    /// </summary>
    [DataMember]
    public DateTime StartTime { get; set; }

    /// <summary>
    /// Gets or sets the End Time of the Build Step
    /// </summary>
    [DataMember]
    public DateTime? EndTime { get; set; }

    /// <summary>
    /// Gets or sets the Duration of the build Step
    /// </summary>
    [DataMember]
    public TimeSpan? Duration { get; set; }
}

```

The implementation of the IBuildReports is shown as follows:

Code 4 – Build duration report TFS 2012 implementation

```

/// <summary>
/// The BuildReports class provides functionality for all Build Reports created from Team Foundation
/// Server's service

```

```

/// </summary>
public class BuildReports : IBuildReports
{
    /// <summary>
    /// Returns the Build Step Duration report for a given build
    /// </summary>
    /// <param name="projectCollectionUri">The Project collection Uri</param>
    /// <param name="teamProjectName">The Team Project Name</param>
    /// <param name="buildDefinitionName">The build definition name for the build for which the report
    /// should return the build duration. Only the latest build is extracted.</param>
    /// <param name="reportOption">An enumeration that determines which steps should be
    /// returned</param>
    /// <returns>A list of build steps with their respective duration</returns>
    public IEnumerable<BuildStep> GetBuildStepDurationReport(string projectCollectionUri,
        string teamProjectName, string buildDefinitionName,
        BuildStepDurationReportOption reportOption)
    {
        ValidateBuildStepDurationParameters(projectCollectionUri, teamProjectName,
            buildDefinitionName);

        var buildCollections = new Collection<BuildStep>();
        var teamCollection = GetTeamCollection(projectCollectionUri);

        var buildserver = teamCollection.GetService<IBuildServer>();
        var buildDefinitionSpec = buildserver.CreateBuildDefinitionSpec(teamProjectName,
            buildDefinitionName);

        var buildDetails = buildserver.QueryBuilds(buildDefinitionSpec);
        if (buildDetails != null && buildDetails.Count() > 0)
        {
            var build = buildDetails.OrderByDescending(s => s.StartTime).FirstOrDefault();
            build.Information
                .GetSortedNodes(new BuildInformationComparer())
                .ForEach(item => AddBuildInformation(buildCollections, item));
        }
        return buildCollections;
    }

    private static void ValidateBuildStepDurationParameters(string projectCollectionUri,
        string teamProjectName, string buildDefinitionName)
    {
        if (string.IsNullOrEmpty(projectCollectionUri))
        {
            throw new ArgumentNullException("projectCollectionUri");
        }

        if (string.IsNullOrEmpty(teamProjectName))
        {
            throw new ArgumentNullException("teamProjectName");
        }

        if (string.IsNullOrEmpty(buildDefinitionName))
        {
            throw new ArgumentNullException("buildDefinitionName");
        }

        if (!Uri.IsWellFormedUriString(projectCollectionUri, UriKind.Absolute))
        {
            throw new ArgumentException("Malformed Uri", "projectCollectionUri");
        }
    }

    private static TfsTeamProjectCollection GetTeamCollection(string projectCollectionUri)
    {
        var teamCollection = TfsTeamProjectCollectionFactory.GetTeamProjectCollection(
            new Uri(projectCollectionUri.ToString(), UriKind.Absolute));
        teamCollection.EnsureAuthenticated();
        return teamCollection;
    }

    private static void AddBuildInformation(Collection<BuildStep> collection,
        IBuildInformationNode buildInformation)
    {

```

```
if (buildInformation.Fields.ContainsKey("StartTime") &&
    buildInformation.Fields.ContainsKey("FinishTime") &&
    buildInformation.Fields.ContainsKey("DisplayText"))
{
    collection.Add(new BuildStep()
    {
        Name = buildInformation.Fields["DisplayText"],
        StartTime = DateTime.Parse(buildInformation.Fields["StartTime"]),
        EndTime = DateTime.Parse(buildInformation.Fields["FinishTime"]),
        Duration = DateTime.Parse(buildInformation.Fields["FinishTime"]) -
            DateTime.Parse(buildInformation.Fields["StartTime"])
    });
}

internal class BuildInformationComparer : IComparer<IBuildInformationNode>
{
    public int Compare(IBuildInformationNode itemToCompare,
        IBuildInformationNode itemToCompareWith)
    {
        if (itemToCompare.Id > itemToCompareWith.Id)
        {
            return 1;
        }
        else if (itemToCompare.Id == itemToCompareWith.Id)
        {
            return 0;
        }
        return -1;
    }
}
```

Service Installation

1. Copy the "_Published" folder of the service to your web server.
2. Create a new Website or Virtual Directory and set it to point to the "_Published" folder.
3. Create a new application pool and set it to run with an account that has the following permissions on the team Project.
 - Delete Builds
 - Update Build Information
 - View Builds
 - View Build Definition
4. Make sure that the option "Enable 32-Bit Applications" on the Application Pool is set to true.
5. Your service is now ready for use.

Build Duration Over Time Report

Context	<p>If you want to optimize your team build and find out how long builds take, Visual Studio's Build log is sometimes difficult to use. Visual Studio doesn't make it easier by making you scroll through endless series of logs.</p> <p>The report detailed here provides a useful alternative by showing you all the builds in the Team Collection in one report with the build durations.</p>
Version	TFS 2012
Category	Health reporting

Table 5 – Build duration over time report

Build Duration Over Time							
Build Definition Name	Build Number	Build Duration	Start Time	Finish Time	Drop Location Root	Build Controller	Build Agent
CI	CI20131104.2	1.0	11/4/2013 9:18:06 AM	11/4/2013 9:19:59 AM	\\MPTFS2012 \Drops	mptfs2012 "Controller\	mptfs2012 - Agent1
CI	CI20131104.1	1.0	11/4/2013 9:15:34 AM	11/4/2013 9:17:27 AM	\\MPTFS2012 \Drops	mptfs2012 "Controller\	mptfs2012 - Agent1
CI	CI20131011.1	3.0	10/11/2013 9:45:57 PM	10/11/2013 9:49:35 PM	\\MPTFS2012 \Drops	mptfs2012 "Controller\	mptfs2012 - Agent1

Figure 3 – Build duration over time report

Prerequisites

- The user should have access to the TFS_<CollectionName> database, where <CollectionName> is the name of the Team collection.

Sample

The query can be extended to a date range to see the results for a time period.

Here's a brief account of each table used in the report:

- tbl_Build – The table stores the record for each team build.
- tbl_BuildQueue – The table stores information of when a particular build got queued.
- tbl_BuildDefinition – The build definition table stores details about the created build definition.
- tbl_BuildController – The build controller table store details about the build controller.
- tbl_BuildInformation – The build information table contains information about build with respect to different element of the build. For example, for each build, there is a build information record for Build Message, Test Impact, etc.
- tbl_BuildInformationType - The types of information stored in the Build Information table are contained in the tbl_BuildInformationType table.

Code 5 – Build Duration Report – TFS 2012

```
SELECT REPLACE(REPLACE(REPLACE(bd.DefinitionName, '\\', ''), '>', ''), '"', '') AS BuildDefinitionName,
       REPLACE(REPLACE(REPLACE(bld.BuildNumber, '\\', ''), '>', ''), '"', '') AS BuildNumber,
       CAST(DATEDIFF(SECOND, bld.StartTime, bld.FinishTime)/60 AS Decimal(5,1)) AS BuildDuration,
       bld.[StartTime], bld.[FinishTime], [DropLocationRoot], bc.DisplayName AS BuildController
       , RA.Field.value('(/Value)[1]', 'varchar(100)') AS BuildAgent
FROM [dbo].[tbl_Build] bld
INNER JOIN tbl_BuildDefinition bd WITH(NOLOCK) ON bld.DefinitionId = bd.DefinitionId
INNER JOIN tbl_BuildInformation bi WITH(NOLOCK) ON bi.BuildId = bld.BuildId
INNER JOIN tbl_BuildController bc WITH(NOLOCK) ON bc.ControllerId = bld.ControllerId
INNER JOIN tbl_BuildInformationType bty WITH(NOLOCK) ON bi.NodeType = bty.NodeType
    cross apply bi.Fields.nodes('/Fields/Field') as RA(Field)
WHERE RA.Field.value('(/@Name)[1]', 'varchar(100)') = 'ReservedAgentName'
ORDER BY StartTime DESC
```

Build Status Over Time Report

Context	<p>If you want to optimize your team build and find out how long builds take, Visual Studio's Build log is sometimes difficult to use. Visual Studio does not make it easier by making you scroll through endless series of logs.</p> <p>The report detailed here provides a useful alternative by showing you all the builds in the Team Collection in one report with the build start time, end time and build durations. The query can be extended to a date range to see the results for a period.</p>
Version	TFS 2012
Category	Health reporting

Table 6 – Build status over time report

Build Definition	Description	Queued Time UTC	Build Number	Build Start Time UTC	Build Finish Time UTC	Queue Time Seconds	Build Controller	Build Agent
Tailspin Toys Iteration 2		3/1/2010 8:01:02 AM	Tailspin Toys Iteration 220100318.1	3/1/2010 8:01:02 AM	3/1/2010 8:02:46 AM	104	WIN\GSGMUJTSB * Controller	WIN\GSGMUJTSB - Agent1
Tailspin Toys Iteration 2		3/1/2010 8:02:49 AM	Tailspin Toys Iteration 220100318.2	3/1/2010 8:02:49 AM	3/1/2010 8:04:16 AM	87	WIN\GSGMUJTSB * Controller	WIN\GSGMUJTSB - Agent1
Tailspin Toys Iteration 2		3/1/2010 4:35:51 PM	Tailspin Toys Iteration 220100318.5	3/1/2010 4:35:51 PM	3/1/2010 4:38:23 PM	152	WIN\GSGMUJTSB * Controller	WIN\GSGMUJTSB - Agent1
Tailspin Toys Iteration 2		3/1/2010 4:28:45 PM	Tailspin Toys Iteration 220100318.4	3/1/2010 4:28:45 PM	3/1/2010 4:30:54 PM	129	WIN\GSGMUJTSB * Controller	WIN\GSGMUJTSB - Agent1
Tailspin Toys Iteration 2		3/1/2010 10:50:31 PM	Tailspin Toys Iteration 220100318.5	3/1/2010 10:50:31 PM	3/1/2010 10:51:56 PM	85	WIN\GSGMUJTSB * Controller	WIN\GSGMUJTSB - Agent1
Tailspin Toys Iteration 2		3/1/2010 2:42:56 PM	Tailspin Toys Iteration 220100318.3	3/1/2010 2:42:56 PM	3/1/2010 2:45:21 PM	145	WIN\GSGMUJTSB * Controller	WIN\GSGMUJTSB - Agent1
Tailspin Toys Iteration 2		3/1/2010 4:39:24 PM	Tailspin Toys Iteration 220100318.6	3/1/2010 4:39:24 PM	3/1/2010 4:42:25 PM	181	WIN\GSGMUJTSB * Controller	WIN\GSGMUJTSB - Agent1

Parameter Values
 Start Date: 3/1/2010
 Finish Date: 3/1/2010
 Build Definition Name: Tailspin Toys Iteration 2
 Display Option: Absolute Scale

Generated: 10/27/2013 1:15:50 PM
 By: VSALMFAAdministrator
 Date Updated: 10/27/2013 12:28:12 PM

Figure 4 – Build status over time report

Prerequisites

- The user should have access to the TFS_<CollectionName> database, where <CollectionName> is the name of the Team collection.

Sample

The following code works for TFS 2012. The build queue table (tbl_BuildQueue) does not contain historical data past 24 hours. If you want to keep more data than you would need to create a separate table in a different database.

You should not create any tables, stored procedures or views in the TFS databases themselves. If you are going to maintain historical data then create a separate database for this.

- tbl_Build – The table stores the record for each team build.
- tbl_BuildQueue – The table stores information of when a particular build got queued.
- tbl_BuildDefinition – The build definition table stores details about the created build definition.
- tbl_BuildController – The build controller table store details about the build controller
- tbl_BuildInformation – The build information table contains information about build with respect to different element of the build. For example, for each build, there is a build information record for Build Message, Test Impact, etc.
- tbl_BuildInformationType - The types of information stored in the Build Information table are contained in the tbl_BuildInformationType table

Code 6 – Build Duration Report – TFS 2012

```

SELECT REPLACE(REPLACE(REPLACE(bd.DefinitionName, '\', ' '), '>', ' '), '"', '') AS BuildDef
, bd.[Description]
--, bq.DateCreated
, bq.QueueTime AS 'Queued Time UTC'
, REPLACE(REPLACE(REPLACE(b.BuildNumber, '\', ' '), '>', ' '), '"', '') AS BuildNumber
, b.StartTime AS 'Build Start Time UTC'
, b.FinishTime AS 'Build Finish Time UTC'
, DATEDIFF(SECOND, bq.QueueTime, b.StartTime) AS 'Queue Time Seconds',
bc.DisplayName AS BuildController,
RA.Field.value('(/Value)[1]', 'varchar(100)') AS BuildAgent
FROM tbl_Build b WITH(NOLOCK)
INNER JOIN tbl_BuildQueue bq WITH(NOLOCK) ON b.BuildId = bq.BuildId
INNER JOIN tbl_BuildDefinition bd WITH(NOLOCK) ON b.DefinitionId = bd.DefinitionId
INNER JOIN tbl_BuildController bc WITH(NOLOCK) ON bc.ControllerId = b.ControllerId
INNER JOIN tbl_BuildInformation bi WITH(NOLOCK) ON bi.BuildId = b.BuildId
INNER JOIN tbl_BuildInformationType bity WITH(NOLOCK) ON bi.NodeType = bity.NodeType
cross apply bi.Fields.nodes('/Fields/Field') as RA(Field)
WHERE RA.Field.value('(/@Name)[1]', 'varchar(100)') = 'ReservedAgentName' AND
bity.TypeName='AgentScopeActivityTracking'
    
```


Mean Time Between Failure (MTBF) Report

Context	<p>The MTBF report, or Mean Time between Failure report, details the average elapsed time a problem is found in the production environment to the moment the next problem was found.</p> <p>In terms of TFS, the occurrence of a problem is tracked by creating of a work item type (WIT) named "Bug." Therefore, to create such a report, the query would look for work item types "Bug." We will use the Tfs_Warehouse database, which stores all information in an approximate star schema, to retrieve the information. The database has a dimension table called DimWorkItem that stores all the work items. However, since it is a dimension table, all changes in any field in the work item results in a new row in this table. Fortunately, the Tfs_Warehouse database also contains a view on the DimWorkItem dimension table that returns the current state of each work item. The view is called CurrentWorkItemView and it is the one that we will be using in our query.</p> <p>Now that we know where the work item information is to be retrieved from, the next factor to consider is to how to determine the time elapsed between the creation of two bugs. The field Microsoft_VSTS_Common_ActivatedDate contains the date/time of when the work item was created so we will use that. If we can sort the results and compute the difference between each adjacent rows in the query, then we have the desired information. The ROW_NUMBER and ORDER clause in SQL Server provides the required functionality.</p> <p>The final piece of the jigsaw is to filter the work items. Because the elapsed time is viewed in historical context, we'll filter the work item to include only the work items that are closed for a specific team project. To do this, we'll filter on the System_State field and the ProjectPath field.</p>
Version	TFS 2012
Category	WIT reporting

Table 7 – MTBF report

Mean Time Between Failures

Week	MTBF in hours
46/2008	449
13/2009	351
23/2009	57
39/2009	170
42/2009	50
49/2009	56
52/2009	87
07/2010	11
10/2010	2
13/2010	0
20/2010	3
23/2010	7
26/2010	2
29/2010	1
36/2010	0
39/2010	0
42/2010	0

Figure 5 –MTBF example

Prerequisites

- The user should have access to the Tfs_Warehouse database.

Sample

CurrentWorkItemView – The view retrieves current state of each work item from the dimension table DimWorkItem.

The query fetches all work items of type “Bug” from the given team project with a state of “Closed” and ordered by Activation Date. The next challenge is to compute the difference between two adjacent records. To do that, it uses the ROWNUMBER AND ORDER clause to give a unique number to each result row. It then joins the result on adjacent rows and computes the difference between the Activated Date of adjacent work items and then using the AVG aggregate function determines the average number of hours.

Code 7 – MTBF Report

```
DECLARE @ProjectPath varchar(255);
-- Using this SELECT as just an example to get the latest build to view the details of
--SELECT @ProjectPath = 'DefaultCollection\TeamProjectName';
SELECT
    RIGHT('0' + CAST(DatePart(wk,CurrentBug.Date) as Varchar) + '/' +
    Cast(DatePart(yyyy,CurrentBug.Date) as Varchar), 7) as Week,
    AVG(DateDiff(hh, FormerBug.Date, CurrentBug.Date)) as [MTBF in hours]
FROM
    (SELECT (
        ROW_NUMBER()
        OVER(ORDER BY Microsoft_VSTS_Common_ActivatedDate asc) + 1) as ID,
        Microsoft_VSTS_Common_ActivatedDate as Date
    FROM
        dbo.CurrentWorkItemView WITH (NOLOCK)
    WHERE
        System_WorkItemType = 'Bug'
        AND System_State = 'Closed'
        and ProjectPath like @ProjectPath
    ) as FormerBug

    INNER JOIN

    (SELECT (
        ROW_NUMBER()
        OVER(ORDER BY Microsoft_VSTS_Common_ActivatedDate asc)) as ID,
        Microsoft_VSTS_Common_ActivatedDate as Date
    FROM
        dbo.CurrentWorkItemView WITH (NOLOCK)
    WHERE
        System_WorkItemType = 'Bug'
        AND System_State = 'Closed'
        and ProjectPath like @ProjectPath
    ) AS CurrentBug
    ON FormerBug.ID = CurrentBug.ID

GROUP BY
    DatePart(wk, CurrentBug.Date),
    DatePart(yyyy, CurrentBug.Date)
```

Mean Time To Resolution (MTTR) Report

Context	<p>MTTR is the average time elapsed from the moment a problem is found in the production environment to the moment that the problem is fixed. To track the MTTR, determine the data you need and how to extract it from TFS. In the MSF for Agile process template, problems found in production are represented by the Bug work item type, so you will filter on that work item type and onTeam Project.</p> <p>Not all the bugs stored in the TFS team project are useful for tracking the MTTR. Only those that have been fixed in production are. You also need to filter by the team project as well as two additional fields to calculate the MTTR. The first field is the moment when the bug is discovered in the production environment. The second field is when the bug is fixed and its state becomes Closed. In the MSF for Agile template, this value is automatically stored in the Closed Date field.</p> <p>You must also decide about how to present the data in the final report. MTTR is calculated as the time elapsed between the Closed Date value and the Activated Date value. However, the report would be more readable if the values were adjusted to a units of time and by grouping the results.</p>
Version	TFS 2012
Category	WIT reporting

Table 8 – MTTR report

Mean Time To Resolution	
Week	MTTR in hours
44/2013	621
43/2013	477
42/2013	379
41/2013	500
40/2013	437
39/2013	415
38/2013	545
37/2013	648
36/2013	562
35/2013	504
34/2013	867
33/2013	981
32/2013	509
31/2013	702
30/2013	1290
29/2013	1375
28/2013	1731
27/2013	615
26/2013	488
25/2013	708
24/2013	873
23/2013	743
22/2013	542
21/2013	647
20/2013	565

Figure 6 – MTTR report example

Prerequisites

- The user should have access to the TFS_Warehouse database.

Sample

Code 8 – MTTR Report

```

DECLARE @ProjectPath varchar(255);
-- Using this SELECT as just an example to get the latest build to view the details of
--SELECT @ProjectPath = 'DefaultCollection\TeamProjectName';
SELECT
    -- We are adding a trailing '0' for 1-digit week numbers,
    -- so we use the RIGHT function to remove it if was not
    -- needed because the week number had 2 digits.
    RIGHT
    (
        '0'
        -- Number of the week inside the year
        + CAST(DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate])
              AS VARCHAR)
        -- Separator
        + '/'
        -- Year
        + CAST(DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate])
              AS VARCHAR)
        -- Maximum length of the returned value (format: WW/YYYY)
        , 7
    ) AS [Week]
    -- Mean Time To Recover, in hours, per each week
    , AVG(
        DATEDIFF(hh, [Microsoft_VSTS_Common_ActivatedDate],
                  [Microsoft_VSTS_Common_ClosedDate])
    ) AS [MTTR in hours]
FROM [dbo].[CurrentWorkItemView] WITH (NOLOCK)
WHERE [ProjectPath] like @ProjectPath = '\DefaultCollection\Lab04-Metrics'
AND [System WorkItemType] = 'Bug'
AND [System_State] = 'Closed'
GROUP BY
    -- Grouping by week and year to calculate the average
    DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate]),
    DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate])
ORDER BY
    -- Ordering by year and week number
    DATEPART(yyyy, [Microsoft_VSTS_Common_ClosedDate]) DESC,
    DATEPART(wk, [Microsoft_VSTS_Common_ClosedDate]) DESC

```

Time Tracking Report

Context	The time tracking report displays each revision of the work item and its impact on the work remaining for that work item. The idea is that as people update work items with parts of work that they have done, they will keep updating the remaining work field. The report will use this query to track how much work is remaining.
Version	TFS 2010, TFS 2012
Category	WIT reporting

Table 9 – Time tracking report

Home > Time Tracking Report

Project Name: [View Report](#)

1 of 1 100% Find | Next

Time Tracking Report

Title	Changed By	Changed Date	Remaining Work	Completed Diff
Create generic configuration management functionality.	Hamid Shahid (NMQA Limited)	10/12/2013 8:28:25 PM	75	20
	Hamid Shahid (NMQA Limited)	10/12/2013 8:34:25 PM	55	15
	Hamid Shahid (NMQA Limited)	10/12/2013 8:35:19 PM	40	20
	Hamid Shahid (NMQA Limited)	10/12/2013 8:36:08 PM	20	20

Figure 7 – Time tracking report example using Scrum 2.2

Home > Custom Reports > Time Tracking Report

Project Name

1 of 1 100% Find | Next

Time Tracking Report

Title	Changed By	Changed Date	Original Estimate	Remaining Work	Completed Work	Completed Diff
Create a generic reporting module						
	Hamid Shahid (NMQA Limited)	10/12/2013 1:38:37 AM	60	60	0	
	Hamid Shahid (NMQA Limited)	10/12/2013 1:38:41 AM	60	55	5	55
	Hamid Shahid (NMQA Limited)	10/12/2013 1:38:46 AM	60	45	15	40
	Hamid Shahid (NMQA Limited)	10/12/2013 1:38:53 AM	60	17	15	2
Create a sample reporting solution						
	Hamid Shahid (NMQA Limited)	10/12/2013 1:39:29 AM	30	30	0	
	Hamid Shahid (NMQA Limited)	10/12/2013 1:39:40 AM	30	22	8	22
	Hamid Shahid (NMQA Limited)	10/12/2013 1:39:50 AM	30	15	15	7
	Hamid Shahid (NMQA Limited)	10/12/2013 1:39:53 AM	30	10	20	-5
	Hamid Shahid (NMQA Limited)	10/12/2013 1:39:58 AM	30	0	30	-20
Create Empty Solutions						
	Hamid Shahid (NMQA Limited)	10/12/2013 1:00:27 AM	8			
	Hamid Shahid (NMQA Limited)	10/12/2013 1:03:55 AM	8			
	Hamid Shahid	10/12/2013	8		13	

Figure 8 – Time tracking report example using MSF Agile

Prerequisites

The user should have access to TFS_Warehouse database.

Sample

Core parameters

This is the list of core parameters needed to drive the report. They should not be deleted.

Parameter	Use
ProjectName	Used to filter the work items. The report displays work items for only the team project with the given name.

Table 10 – Time tracking report parameter

The parameter is set as follows:

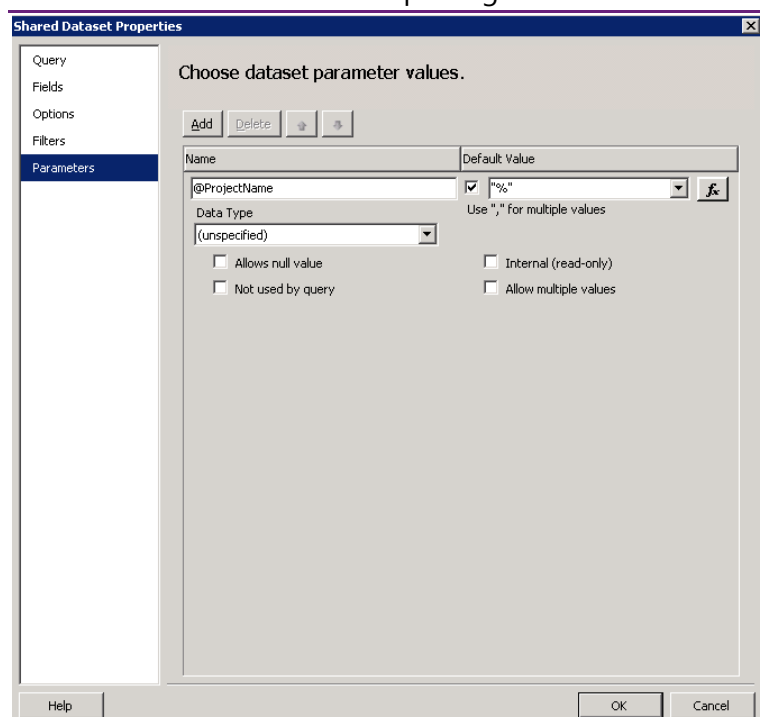


Figure 11 – Report Parameter

Data sources

The report only contains one data source **Tfs_Warehouse** and should be set to the Tfs_Warehouse database.

Walkthrough

Step	Instructions
1 Create report project ☐ - Done	<ul style="list-style-type: none"> Start BIDS (SQL Server Business Intelligence Development Studio). Create a new Report Server Project MyCustomReport.
2 Add Datasource ☐ - Done	<ul style="list-style-type: none"> In solution explorer, right click on DataSource, Add New Datasource and name it Tfs_Warehouse Provide the connection string and credentials to read the Tfs_Warehouse database.
3 Add Dataset ☐ - Done	<ul style="list-style-type: none"> In solution explorer, right click on DataSets, Add New DataSet. Set the name of the data set as TimeTrackingDataset. Select DataSource Tfs_Warehouse. In the Text Area for Query, copy and paste the following SQL code: <pre>SELECT [System_Id] AS [Id],[System_Rev] AS [Rev], [System_ChangedBy] AS [Changed By], [Microsoft_VSTS_Scheduling_OriginalEstimate] AS [Original Estimate], [Microsoft_VSTS_Scheduling_RemainingWork] AS [Remaining Work], [Microsoft_VSTS_Scheduling_CompletedWork] AS [Completed Work], [System_Title] AS [Title],[System_AssignedTo] AS [Assigned To], [System_ChangedDate] AS [Changed Date], [ProjectNodeName] AS [Team Project], [AreaName],[AreaPath], [IterationName], [IterationPath] ,</pre>

TFS Practical Reporting Guide - Part 2: Data Warehouse – Practical Reports

Step	Instructions
	<pre> [Microsoft_VSTS_Scheduling_RemainingWork] - (Select Top 1 IsNull([Microsoft_VSTS_Scheduling_CompletedWork],0) FROM [Tfs_Warehouse].[dbo].[WorkItemHistoryView] WITH(NOLOCK) WHERE [System_Id] = WIHV.[System_Id] AND [System_Rev] = WIHV.[System_Rev] - 1) as [Completed Diff] FROM [Tfs_Warehouse].[dbo].[WorkItemHistoryView] WIHV WITH(NOLOCK) WHERE /** Filter team project */ [ProjectNodeName] LIKE @ProjectName /** Filter correction entries */ AND [RevisionCount] IS NOT NULL /** Filter empty entries */ AND ([Microsoft_VSTS_Scheduling_OriginalEstimate] IS NOT NULL OR [Microsoft_VSTS_Scheduling_RemainingWork] IS NOT NULL OR [Microsoft_VSTS_Scheduling_CompletedWork] IS NOT NULL) ORDER BY ID, Rev </pre> <ul style="list-style-type: none"> Click on the Parameters option and select the check box for Default Value. Set the default value as "%". Click OK.
4 Add report <input type="checkbox"/> - Done	<ul style="list-style-type: none"> In Solution Explorer, right-click Report, Add existing report and browse to the report TimeTrackingReport.rdl.
5 Set as start report for project <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Right-click the project node and select Properties. Under Debug/Startitem select the report.
6 Test report <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Press F5 to build and test the report.

Table 12 – Time tracking report

Custom Report Service

Team Foundation Server creates an OLTP database for each of its project collections. The database for default team project collection is called `Tfs_DefaultCollection`. This is where TFS stores all “active” data. Data from this database is fed first into the star schema `Tfs_Warehouse` database and then to the `Tfs_Analysis` OLAP cube.

Even though reporting from `Tfs_Warehouse` and `Tfs_Analysis` databases is supported, Microsoft does not recommend writing reports directly from the `Tfs_DefaultCollection` database. The reason is that schema of the OLTP database are often changed by new TFS releases and updates.

However, the TFS API provides an alternative way of fetching data from `Tfs_DefaultCollection` database. You can write your own services that use the TFS API and provide an alternative way of getting data.

This guide is shipped with a sample WCF service. The service uses TFS API to get information about builds and build steps from OLTP database.

Pre-requisites



To set up the custom report service, you will need a machine with the following Microsoft Windows features installed on it:

- Microsoft Internet Information Server (IIS) v 7.0 or v. 8.0.
- Microsoft .NET Framework 4.

Security Requirement

- You will need to have a service account such that the account that has been added to the “Builders” group in TFS. The Builders group is created for every newly created Team Project Collection.

Walkthrough

Step	Instructions
1 Installation  - Done	<ul style="list-style-type: none"> • Copy the Teamreports\TeamReportService folder from the companion sample code to the web server where you are looking to deploy the TFS Custom Service. You would use the TFS Application layer server or some other machine that can call TFS API. • Open Internet Information Services and create an application pool called TFSReports. • Select .Net Framework v4.0.30319 for .NET Framework version and Integrated for pipeline mode. • Click OK to create the application pool. Right-click it and select Advanced Settings. • Click Identity and set the account to the service account as specified in security requirement above. • Create a new website called TfsReports. Select the newly created TFSReports application pools as its application pool and point it to the copied TeamReportsService directory. • For the binding of the newly created website, either give it a unique DNS name or assign it to a unique port on the server. Click OK. Your service is now deployed and ready to be called.
2 Understand Signature  - Done	<ul style="list-style-type: none"> • Your TFS Custom Report Service can now be called from any client that can make SOAP requests. The sample service contains one service called BuildService with a one operation GetBuildStepDurationReport. The signature of the operation is as follows <pre> /// <summary> /// Returns the Build Step Duration report for a given build /// </summary> /// <param name="projectCollectionUri">The Project collection Uri</param> /// <param name="teamProjectName">The Team Project Name</param> /// <param name="buildDefinitionName">The build definition name for the build for which the report should return the build duration. Only the latest build is extracted.</param> /// <param name="reportOption">An enumeration that determines which steps should be returned</param> </pre>

TFS Practical Reporting Guide - Part 2: Data Warehouse – Custom Report Service

Step	Instructions
	<pre>/// <returns>A list of build steps with their respective duration</returns> [OperationContract] IEnumerable<BuildStep> GetBuildStepDurationReport(string projectCollectionUri, string teamProjectName, string buildDefinitionName, BuildStepDurationReportOption reportOption);</pre> <ul style="list-style-type: none"> It takes the Build Id as a parameter and returns a list of all builds.
3 Understand Operations <input type="checkbox"/> - Done	<ul style="list-style-type: none"> The operation takes in parameters such as the Team Project Collection URL, the Team Project Name, build definition name and an enumeration that allows the caller to specify whether it needs all steps, the fastest ten or the slowest ten steps. The result is an enumeration of BuildStep objects, which contains properties such as Build Step Name, Start Time, Finish Time and Duration. <pre>/// <summary> /// The BuildStep class encapsulates a single Build Step in a build report /// </summary> [DataContract] public class BuildStep { /// <summary> /// Gets or sets the name of the Build Step /// </summary> [DataMember] public string Name { get; set; } /// <summary> /// Gets or sets the Start Time of the Build Step /// </summary> [DataMember] public DateTime StartTime { get; set; } /// <summary> /// Gets or sets the End Time of the Build Step /// </summary> [DataMember] public DateTime? EndTime { get; set; } /// <summary> /// Gets or sets the Duration of the build Step /// </summary> [DataMember] public TimeSpan? Duration { get; set; } }</pre> <ul style="list-style-type: none"> The operation returns a list of steps with their execution duration of the latest build.. The report is useful for build managers who want to diagnose the slower running parts of their reports.
4 Client Application Proxy <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Using the service can be demonstrated using the WcfTestClient.exe application that is included with Visual Studio. To create the client, following these steps: Create a new Windows console application in Visual Studio. Right-click service reference and click Add Service Reference. Type in the service URL http://TFSCustomReports/BuildService.svc and click GO. Select BuildReports, type in a suitable namespace name and click OK.

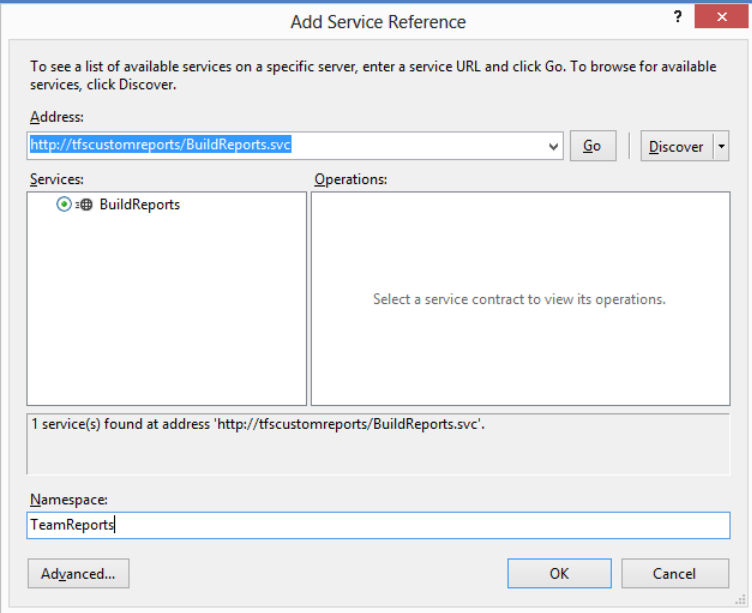
Step	Instructions
	 <ul style="list-style-type: none"> Click OK. This will generate a proxy class for the service.
5 Client Code <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Now, create an instance of the proxy class and call the proxy class. The final listing is like this: <pre>using (var client = new BuildReportsClient()) { var buildSteps = client.GetBuildStepDurationReport("*TeamProjectCollectionUri", "TeamProjectName", "BuildDefinitionName", BuildStepDurationReportOption.All); foreach (var step in buildSteps) { Console.WriteLine("Build Step {0} - Duration {1}", step.Name, step.Duration); } }</pre>
6 Run client <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Build and run the client application by pressing F5.

Table 13 – Custom report service walkthrough

NOTE

The sample service and client is included in the companion sample code included with this guide.

Data Adapters and the TFS Warehouse

WARNING

This section is still under construction, which will be updated shortly.

We apologize for any inconvenience caused, but wish to ensure we deliver the best quality guidance when ready.



Microsoft uses [warehouse adapters](#)³⁰ to add data to the TFS Data Warehouse. These adapters are used to take data from the operational (transactional) tables and format it into dimension and fact table form. You can create your own adapter to augment the data in the warehouse.

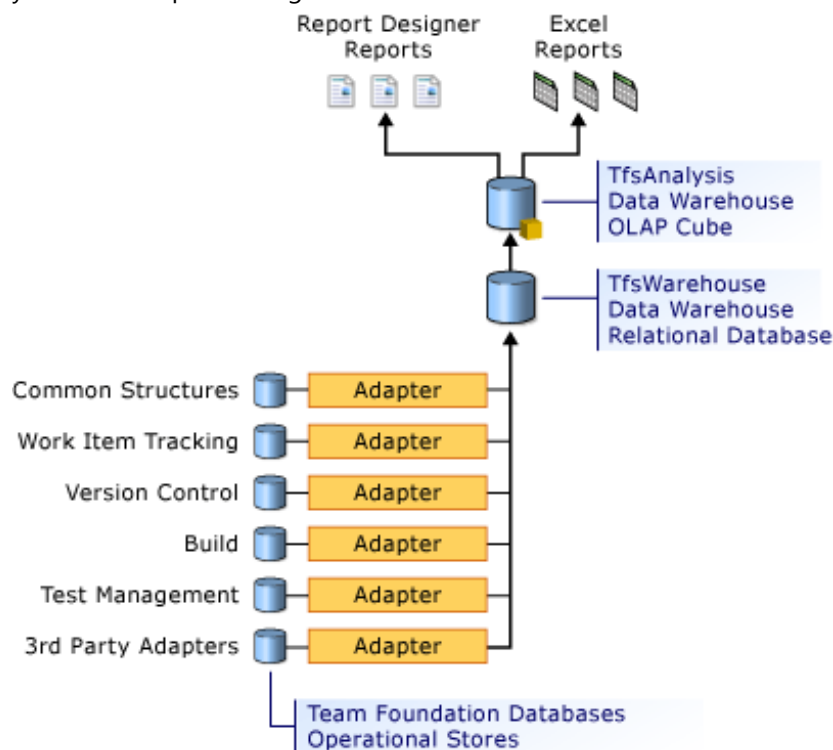


Figure 9 - Data Warehouse architecture

³⁰ [http://msdn.microsoft.com/en-us/library/vstudio/ms244687\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/vstudio/ms244687(v=vs.110).aspx)

You can find information about how to build an adapter on earlier versions of TFS in these posts from [Mattias Sköld](#)³¹, [Steve St. Jean](#)³² and [Steven Wright](#)³³. If you need something more prescriptive, there is downloadable [code](#)³⁴ and a walkthrough on the Microsoft [website](#)³⁵.

TFS Warehouse Adapter Basics

Implementing an Adapter

Two interfaces are implemented when you get data from the operational tables and insert it into the TFS data warehouse: `IWarehouseAdapter` and `IDataStore`. The `IWarehouse` interface is used by the TFS data warehouse service to call the five core adapters on a scheduled basis for data changes. The `IDataStore` interface is used by the TFS data warehouse service to interact with the TFS data warehouse itself.

The `IWarehouseAdapter` has four methods:

- `Initialize`: creates objects that communicate with the operational store and the warehouse.
- `MakeSchemaChanges`: creates a new dimension attribute in the warehouse. This attribute contains the policy override description.
- `MakeDataChanges`: is called each time you update the warehouse. In this adapter, the method checks all changesets made since the last warehouse update. If a changeset contains a description of the policy override, it calls the `SavePolicyOverrideCommentDimAttribute` method, which saves the description in the warehouse.
- `Cancel`: when you want to stop the adapter.

Using the CSharp Assembly Churn sample provided by Microsoft, you will see the best practices of creating a separate dimension table for data you are adding to the TFS warehouse. You need to keep in mind that versions of TFS will be different and you don't want to be tied to their tables since there is no guarantee that they will be the same after a cumulative update or version change.

The sample contains SQL and classes used in the creation of a dimension tables, fact table and stored procedures used in the TFS data warehouse.

An implementation of using an adapter to make data changes is found in the `CSharpAssemblyCodeChurnSampleAdapter` class. This class has the `ProcessChangesets` method that is used to get the assembly information for files and add that to the TFS data warehouse.

Deploying the Adapter

A best practices approach would be testing this out in a non-production environment. You might consider using a Brian Keller VM if you only have one environment.

When you have built your adapter you need to remote into your TFS application tier.

- Copy the assembly into the warehouse plugins folder
 - **"C:\Program Files\Microsoft Team Foundation Server 11.0\Application Tier\Web Services\bin\Plugins"** for TFS 2012
 - **"C:\Program Files\Microsoft Team Foundation Server 12.0\Application Tier\Web Services\bin\Plugins"** for TFS 2013
- Reset IIS.

³¹ <http://mskold.blogspot.com/2009/08/tfs-admin-part-iii-poc-extending-tfs.html>

³² <http://sstjean.blogspot.com/2008/12/tfs-warehouse-adapter-how-to-link.html>

³³ <http://zogamorph.blogspot.com/2008/07/how-i-built-team-foundation-server.html>

³⁴ <http://archive.msdn.microsoft.com/Tfs2010SampleAdapter>

³⁵ [http://msdn.microsoft.com/en-us/library/bb286956\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/bb286956(v=vs.90).aspx)

- Open a browser and go to <https://localhost/tfs/TeamFoundation/Administration/v3.0/WarehouseControlService.aspx?op=GetProcessingStatus> and click **Invoke**.
- Continue doing this until the status is returned as **Idle**.
- Follow through to make sure that this hasn't caused any problems by checking the Windows application event log to confirm there aren't any errors.

As with any code, you should source control it to adjust for any changes in TFS warehouse interfaces.

It is recommended to monitor your TFS data warehouse processing status. Grant Holliday's Cube Processing Status report will provide information on errors that are encountered.

Important Information

Items to consider:

Important schema changes in the warehouse table structure:

- TFS 2010: still used a lot of cascade-delete foreign keys, kills performance
- TFS 2010 SP1: dropped most foreign key relations, adapters need to clean up after themselves
- TFS 2012/2013: hardly any changes to the warehouse, but the operational datastore schema was changed considerably, especially around build

.NET Framework versions, CPU settings for adapters

- TFS 2010: .NET 4, AnyCPU, expect to be loaded in x64 environment
- TFS 2012: .NET 4.5, AnyCPU, expect to be loaded in x64 environment
- TFS 2013: .NET 4.5.1, AnyCPU, expect to be loaded in x64 environment

Minimal SQL Server features

- TFS 2010 (SQL Server 2008)
- TFS 2012 (SQL Server 2008 R2)
- TFS 2013 (SQL Server 2012)
- SQL 2014 hasn't been verified

Assemblies to reference and where you can find them

- Overview of the SK/BK's for existing dimensions and facts
- Debug an adapter (attach Visual Studio to the TFS Job service)
- Triggering an adapter at will (call into the warehouse web service)

In Conclusion

This concludes our adventure into the Team Foundation Server Data Warehouse. We have touched on theory and introduced you to the Data Warehouse and Reporting. We have covered various exercises in the walk-throughs and guided you through the theory, design and practical usage of the TFS Data Warehouse to proactively monitor and detect TFS environmental health issues.

We hope you find it a valuable technology to invest in and that you have found this guide useful.

Sincerely

The Microsoft Visual Studio ALM Rangers



Appendix

A report template to start with

To create your custom report and have it received and delivered as a TFS report there is a common ground you need to meet. To make this as easy and effortless as possible, we've created a basic empty report that you can build on. In this appendix, we'll go through the template and explore how it works.

Walkthrough of the report template

Layout

The report template contains all the layout elements of a standard TFS report;

1. A report description at the top outlines the purpose of the report.
2. Links to other related reports on the right for other reports in this category.
3. The question that the report helps answer.
4. Parameter values used for generating the report. This is useful if you want to print the report.
5. Data updated timestamp to indicate the "freshness" of the data in the report.

[&ReportName] 1 Helps you track recent progress for each user story. Shows each story's remaining, completed, and recently completed work.		Related Reports Related report link 1 Related report link 2 2 Related report link 3
There are no results matching the parameters. Try choosing more general parameter values.		
«Expr»		
Questions This Report Helps Answer • Example question 1 • Exempel question 2 3 How to Use This Report	Parameter Values Iteration: «Expr» Area: «Expr» 4 Recent (Calendar) [:@Period] Display Option: [:@Scaling.Label]	«Expr» «Expr» «Expr» 5 [:@Warning]

Figure 10 – Layout of the report template.

Parameters

The template contains a set of core parameters for driving the report and a set of optional parameters set up to support common reporting scenarios. If you don't use one of the optional parameters, you can simply delete it.

Core parameters

This is the list of core parameters needed to drive the report. They should not be deleted.

Parameter	Use
ExplicitProject	Used to debug/test report
ReportPath	Contains the path to the RDL file, used by dsProjectGuid to get the ProjectGuid
ProjectGuid	Contains the current project Guid
ProjectName	Contains the current project name
IsDashboard	Hides all the standard layout elements if set to true, normally overridden in the URL

Table 14 – Code parameters Remove

Optional parameters

This is the list of optional parameters to support common reporting scenarios,.Deletethem if you don't use them.

Parameter	Use
IterationParam	Contains selected iterations,
AreaParam	Contains selected areas.

Table 15 – Optional parameters.

Data sources

The report only contains one datasource **TFSReports**. TFS Reports is the datasource for connecting to your TFS relational warehouse. For more information about TFS Relational Warehouse and its schema, please refer to <http://msdn.microsoft.com/library/bb649552.aspx>

Datasets

The template contains a set of core datasets for driving the report and a set of optional datasets to support common reporting scenarios. If you don't use one of the optional dataset you can simply delete them and their corresponding parameters.

Core datasets

This is the list of core datasets needed to drive the report. They should not be deleted.

Dataset	Use
dsProjectGuid	Gets the project quid from the current report path
dsLastProcessedTime	Get the last update for the data

Table 16 – Code datasets.

Optional datasets

This is the list of optional datasets

Dataset	Use
dsIteration	Gets all Iterations for the team project
dsArea	Gets all areas for the team project

Table 17 – Optional datasets.

Getting started

This part will guide you through how to start developing a new report using the template.

Follow these steps:

Step	Instructions
1 Create new Report Project <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Start BIDS (SQL Server Business Intelligence Development Studio). Create a new Report Server Project MyCustomReport.
2 Add shared Datasource TfsReportDs <input type="checkbox"/> - Done	<ul style="list-style-type: none"> In Solution Explorer, right-click DataSource, Add New Datasource. Name it TfsReportDs. Provide the connections string and credentials to read the TFS Relational Database.
3 Add and rename the template <input type="checkbox"/> - Done	<ul style="list-style-type: none"> In Solution Explorer, right-click Report, Add existing report. Copy and rename the ALM Rangers Report template to the project folder. Select and add the newly copied report.
4 Set report parameter ExplicitProject <input type="checkbox"/> - Done	<ul style="list-style-type: none"> In Report Data Explorer, right-click ExplicitProject, Parameter Properties. Select Default Values tab. Set the default values to the path of an existing team project: /TfsReports/DefaultCollection/TestProject
5 Set report as start report for the project <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Right-click the project node, Properties. Under Debug/Start item select the report.
6 Test report <input type="checkbox"/> - Done	<ul style="list-style-type: none"> Press F5.

Table 18 – Report Template – Getting Started

Customizing your VM for Analysis Services

Install Analysis Services in tabular mode on Brian Keller's VM

Install SQL Server 2012 Analysis Services

1. Launch [Brian Keller's VM](#) ³⁶ and log on as the Administrator account.
2. Mount the **SQL Server 2012** Installation disc or path to a share with the installation media.
3. Start the installation.
4. On the **Product Key** page click **Next**.
5. Accept the **license terms** and click **Next**.
6. Click **Next** on the **Product Updates** page.
Ignore the warning. We're going to install SP1 afterwards.
7. After the setup support rules run, click **Next**.
8. On the **Setup Role** page select **SQL Server Feature Installation** and click **Next**.
9. On the **feature selection** page select only Analysis Services, Client Tools Connectivity and Client Tools Backwards Compatibility, Management Tools – Basic and Complete and SQL Server Data Tools and click **Next**.
Note that at least for this go around we aren't going to try to install PowerView. That's a more involved process if we ever get there so let's not worry about that right now
10. After the installation rules run, click **Next**.
11. On the **Instance Configuration**, it has to be a named instance so name it "Tabular" and click **Next**.
12. On the **disk space requirements** click **Next**.
13. Under **server configuration** click **Next**.
14. On the **Analysis Services Configuration** page select **Tabular Mode** (the default is Multidimensional and Data Mining Mode).
15. Click **Add Current User** and click **Next** (feel free to add other users but it isn't necessary).
16. On the **Error Reporting** page click **Next**.
17. After the installation configuration rules run, click **Next**.
18. On the ready to install page click **Install**.
19. After the install is complete, click **Close**.

Install SQL Server 2012 x64 SP1

1. Insert the **SQL Server 2012** disc in the VM and execute the x64 SP1 install.

Verify Installation

1. To verify that everything is correct, open **SQL Server Management Server 2012** and connect to **localhost\tabular** and check the properties.

³⁶ <http://aka.ms/VS11ALMVM>

Understanding the Work Item Store Relational Data Model

Each new instance of Team Foundation Server creates the following database on the Data Tier:

- Tfs_Configuration
- Tfs_DefaultCollection (or Tfs_<CollectionName> if your team collection has a different name)
- Tfs_Warehouse
- Tfs_Analysis

The first three databases are OLTP databases created on the SQL Server database service. The Tfs_Analysis is an OLAP database created on the SQL Server Analysis service. The following table provides a brief description of each of database:

Database Name	Type	Description
Tfs_Configuration	Relational database	This database stores catalog of resources and configuration of Team Foundation Server. Each TFS installation has exactly one Tfs_Configuration database.
Tfs_<CollectionName>	Relational database	Each new team collections results in the creation of a new Tfs_<CollectionName> database. This database holds actual data of team projects within the team collection. This is the primary data store that is written to as new work items are created or files are checked in.
Tfs_Warehouse	Relational database	The Tfs_Warehouse is the relational database warehouse of TFS and provides data in the format that is easier for reporting.
Tfs_Analysis	OLAP Cube	The Tfs_Analysis database is the multi-dimensional OLAP cube contains team foundation server's aggregated data.

Table 19 – TFS Data tier databases

Microsoft recommend using the databases Tfs_Warehouse and Tfs_Analysis for reporting. There is ample documentation about the structure and schema of these databases. However, data feeds to Tfs_Warehouse occurs every two hours and the format might not be desirable in certain circumstance. All team collection databases converge in these databases (Tfs_Warehouse and Tfs_Analysis).

This section describes the schema of Tfs_DefaultCollection to assist users who want to use the data in the OLTP database for reporting purposes.

WARNING

Microsoft does not recommend using this database for reporting directly, for these reasons:

- The Tfs_DefaultCollection schema database might change in future versions or updates of TFS.
- Querying the database directly might have an adverse effect on the performance of your TFS instance. It is therefore advisable to use locking hints so that your queries are not locking the database.
- Any accidental change in the data of this database might corrupt your TSF installation. The data in this database should never be change. Always connect to the database with a user that has only read-only access to the database.

To prove the recommendation from Microsoft, when creating a new Team Project (default is on Reporting Services), we only have connectivity to the Data Sources of databases Tfs_Analysis (named TFSReportsDS) and Tfs_Analysis (named TFSOlapReportsDS). In this document, we will look in detail at how data related to work items is stored in the Team Foundation Server database.

Tfs_DefaultCollection Tables

At first sight, the Tfs_DefaultCollection database schema looks somewhat convoluted. There are dozens of tables and it is not very obvious which table contains what data. Because we are concentrating on work items, we start with work items related tables:

[dbo].[WorkItemsAre]

This table is the primary table that store all work items. The table contains the following columns:

Field Name	Description
[PartitionId]	The column can be used to partition the table. Unless partitioning is configured, it is always populated with a value of 1.
[Changed Date]	The date/time when the work item was last changed.
[PersonId]	The ID of the person who created the work item.
[AreaId]	The ID of the Area to which the work item belongs. The field contains foreign key reference to the [xxTree] table
[Rev]	The revision number of the work item. Every time the work item is updated, the number in this field is incremented by 1.
[State]	Text field containing the state of the work item.
[Reason]	Text field containing the reason of the work item.
[Assigned To]	The ID of the user this work item is assigned to.
[Created By]	The ID of the user who created this work item.
[ID]	The ID of the work item.
[Changed Order]	The timestamp field for the record.
[Authorized Date]	The date/time when the work item was last updated.
[Created Date]	The date/time when this work item was created.
[IterationID]	The iteration selected for the work item. The field contains foreign key reference to the [xxTree] table.
[Title]	Text field containing the title of the work item.
[Changed By]	The ID of the user who last changed this work item.
[Work Item Type]	Text field containing the type of work item.

Table 20 – [dbo].[WorkItemsAre] table

In addition to the above fields, the table also contains several fields that start with the name "fld" followed by a number such as "fld10002". These are dynamic fields. They provide the flexibility of modifying process templates. The second part of the field is the ID of the field, whose details are contained in the "fields" table.

For example, the following table returns the name and value of the field fld10002 for all work items:

```
SELECT WorkItems.ID, Fields.Name as FieldName, WorkItems.Fld10002 as FieldValue
FROM [WorkItemsAre] WorkItems
LEFT JOIN [Fields] ON Fields.FldID = 10002
```

[dbo].[WorkItemsLatest]

The table [dbo].[WorkItemsLatest] has the same schema as the table [dbo].[WorkItemsAre], except that it contains an extra field called "Revised Date". This table can be used interchangeably with the table [dbo].[WorkItemsAre].

[dbo].[WorkItemsWere]

The table [dbo].[WorkItemsWere] contains historic data of [dbo].[WorkItemsAre] table. It has the same schema as the [dbo].[WorkItemsAre] table. Every time a change to a work item is made, a new record is created in this table. The new record contains the state of the work item before change.

[dbo].[WorkItemLongTexts]

The html fields in Work Items are stored in a separate table called [dbo].[WorkItemLongTexts]. Every time the html field is updated, a new record is created in this table. However, this table is written to only when there is a change in html field. It means that not all revision numbers preset in the [dbo].[WorkItemsWere] table will have a corresponding record in this table. The schema of the table is detailed here:

Field Name	Description
[PartitionId]	The column can be used to partition the table. Unless partitioning is configured, it is always populated with a value of 1.
[AddedDate]	The date when a value was added to the html field.
[FldID]	The ID of the html file. This key is foreign key referencing the [FldId] field in the fields table.
[ID]	The work item ID for this html field.
[Words]	The text of the html field.
[Changed Order]	The timestamp field for the record.
[Rev]	The revision number of the change made to the html field.
[fHtml]	A Boolean field indicating whether the field is an html field or not.
[IndexedWords]	It is a binary field containing the hash of the Words field.
[WordsDocumentType]	A string field containing the type of data stored in the field. Possible values are .txt, .htm.
[TextID]	It is an identity field and the primary key of the table.
[EndDate]	The End Date is only populated for records, which are not currently active. That is, the field was updated and a new record is created with the new values. For each update to a long field, the EndDate field is populated for the current record and a new record is created.

Table 21 – [dbo].[WorkItemLongTexts] table

[dbo].[WorkItemFiles]

The table [dbo].[WorkItemFiles] contain information of all files attached to work items. The following table details the schema of the [dbo].[WorkItemFiles] table:

Field Name	Description
[PartitionId]	The column can be used to partition the table. Unless partitioning is configured, it is always populated with a value of 1.
[RemovedDate]	The date/time when the attached was deleted from the work item. Only populated for deleted attachments.
[AddedDate]	The date/time when the file was attached to the work item.
[FldID]	The ID of the field in the work item that stores the attached file.
[ID]	The ID of the work item to which the file is attached.
[FilePath]	The field is populated with the GUID from the FileGuid field in the [dbo].[Attachment] table where the file contents are stored.
[ExtID]	An identity field and the primary key of the table.

Field Name	Description
[Comment]	A string field containing the comments written by the user when attached file.
[CreationDate]	The creation date/time of the attached file.
[LastWriteDate]	The last modified date/time of the attached file.
[Length]	The size of the file in bytes.
[Historical Added Date]	The date/time when the work item was saved with the attached file added.
[Historical Removed Date]	The date/time when the work item was saved with the attached file removed.

Table 22 – [dbo].[WorkItemFiles] table

[dbo].[LinksAre]

The table [dbo].[LinksAre] contain information of one or more links

Field Name	Description
[PartitionId]	The column can be used to partition the table. Unless partitioning is configured, it is always populated with a value of 1.
[SourceID]	The ID of the work item from where the link is created.
[TargetID]	The ID of the work item to which the link is created.
[LinkType]	The Link Type ID for the Link. This field is a foreign key referencing the table [dbo].[LinkTypes], which contains information of different types of links that can be created between two work items.
[ReverseLinkType]	The Link Type ID for the Link from the target work item to the source work item. This field is a foreign key referencing the table [dbo].[LinkTypes], which contains information of different types of links that can be created between two work items.
[Created By]	The user ID of the person who has created the link.
[Created Date]	The date/time when the link was created.
[Comment]	The comment entered by the user while creating link.
[Historical Added Date]	The date/time when the work item was saved with the attached file added.

Table 23 – [dbo].[LinksAre] table

[dbo].[LinksWere]

Similar to [dbo].[WorkItemsWere] table, the [dbo].[LinksWere] contain historical information of links between work items. When a link between work items is edited or deleted, a new row is created in the [dbo].[LinksWere] table reading information of the unmodified link from [dbo].[LinksAre] table before the record in the [dbo].[LinksAre] table is either edited or deleted.

[dbo].[WorkItemsDestroyed]

The table [dbo].[WorkItemsDestroyed] contains the list of destroyed work items. When a work item is destroyed, its data is removed from the [dbo].[WorkItemsAre] and [dbo].[WorkItemsWere] table and a new record is created in the [dbo].[WorkItemsDestroyed] table.

Field Name	Description
[PartitionId]	The column can be used to partition the table. Unless partitioning is configured, it is always populated with a value of 1.
[ID]	The ID of the work item that was destroyed.

Field Name	Description
[Changed Order]	The timestamp field of the record.
[Changer ID]	The ID of the user who destroyed the work item.

Table 24 – [dbo].[WorkItemsDestroyed] table**[dbo].[WorkItemLinksDestroyed]**

When a work item linked to other work items either as a source or a target is destroyed, a record is created in the table [dbo].[WorkItemLinksDestroyed] table. The table has the following schema:

Field Name	Description
[PartitionId]	The column can be used to partition the table. Unless partitioning is configured, it is always populated with a value of 1.
[SourceID]	The ID of the source work item.
[TargetID]	The ID of the target work item.
[LinkType]	A string field containing description of the type of link that was destroyed.
[ChangeDate]	The date when the link was destroyed.
[DeletedTimeStamp]	The timestamp of the original link record that was deleted.
[TimeStamp]	The timestamp of the record.

Table 25 – [dbo].[WorkItemLinksDestroyed]