KEYSIGHT TECHNOLOGIES

Developer Migration Guide

Notices

DFARS/Restricted Rights Notice

If software is for use in the performance of a U.S. Government prime contract or subcontract, software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Keysight Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED “AS IS,” AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Keysight Test Automation version 9.9, 2020
Keysight Technologies, Inc.
900 South Taft Avenue, Loveland, CO, 80537-6378 USA

For support, go to the Keysight Test Automation website and click **Contact an Expert**.

Table of Contents

Migrating Plugins from Version 8.x to Version 9.x (OpenTAP)

OpenTAP 9.x contains a number of breaking changes relative to TAP 8.x - hence the change in major version number. This page contains help on how to migrate from TAP 8.x.

TAP -> OpenTAP

TAP has been renamed to OpenTAP. This has several consequences for plugins depending on OpenTAP.

It means the convention changes a bit for naming assemblies and namespaces. We recommend [OrganizationName].OpenTap.[PluginName] for closed-source and OpenTap.[PluginName] for open-source. So a closed-source Keysight plugin would be called Keysight.OpenTap.XSeries. These recommendations has been applied to the OpenTAP core plugins.

Namespace changes

TAP is renamed to OpenTAP, this is also changed in the namespaces. Additionally, the core of TAP (the Engine, CLI, PackageManager and BasicSteps) are being open sourced, so we are removing “Keysight” from the namespace in those parts.

- Keysight.Tap -> OpenTap
- Keysight.Tap.Cli -> OpenTap.Cli
- Keysight.Tap.Package -> OpenTap.Package
- Keysight.Tap.Gui -> Keysight.OpenTap.Gui
- Keysight.Tap.Sdk -> Keysight.OpenTap.Sdk
- Keysight.Tap.Gui.Controls -> Keysight.OpenTap.Wpf
- Keysight.Tap.TimingAnalyzer -> Keysight.OpenTap.TimingAnalyzer
- Keysight.Tap.ResultsViewer -> Keysight.OpenTap.ResultsViewer
- Keysight.Tap.RunExplorer -> Keysight.OpenTap.RunExplorer

File name changes

Filenames have changes as well following the namespace changes above.

- Keysight.Tap.Engine.dll -> OpenTap.dll
- Keysight.Tap.PackageManager.exe -> OpenTap.Package.dll
- Keysight.Tap.Gui.exe -> Editor.exe
- Keysight.Tap.TimingAnalyzer.exe -> TimingAnalyzer.exe
- Keysight.Tap.ResultsViewer.exe -> ResultsViewer.exe
- Keysight.Tap.RunExplorer.exe -> RunExplorer.exe
- Keysight.Tap.Sdk.MSBuild.dll -> Keysight.OpenTap.Sdk.MSBuild.dll (in “Packages” folder)
- Keysight.Tap.Gui.Controls.dll -> Keysight.OpenTap.Wpf.dll (in “PackagesControls” folder)

Package payload location convention

Previously it was the convention that packages put their payload in a subfolder with the same name as the package. In 9.x we change the convention and recommend that packages put payload in a `Packages/<PackageName>/` subfolder. This is also where the package.xml metadata file will now be located (moved from `Package Definitions/<PackageName>.package.xml` to `Packages/<PackageName>/package.xml`).

SwitchMatrixRow/Column

These two specializations of `ViaPoint` was there to ease implementing switch matrices, however they did not support matrices on which several paths could be active at the same time. A new `SwitchMatrixPath` class replaces them.

CLI commands

Support for CLI subcommands has been implemented. An ICliAction can be grouped by using the "Group" property in the DisplayAttribute.

New `tap.exe` usage:

```
tap <command> [<subcommand>] [<args>]
```

This introduces the follow breaking changes:

- `tap create` changed to `tap package create`
- `tap download` changed to `tap package download`
- `tap install` changed to `tap package install`
- `tap uninstall` changed to `tap package uninstall`
- `tap list` changed to `tap package list`
- `tap test` changed to `tap package test`

Running a test plan through the CLI has also changed:

- `Keysight.Tap.Cli MyTestPlan.TestPlan` changed to `tap run MyTestPlan.TestPlan`
- Service mode removed. Use REST-API for similar behavior.

Git Assisted Versioning

`TapVersion`, `GitBranch` and `GitBranchVersion` macros have been deleted. Use `GitVersion` instead.

The Git assisted versioning (using `GitVersion`) scheme has changed. Now a `.gitversion` file is mandatory in the root of the repository. This file defines the version number to be used. Please see the Developer Guide (the 'Git Assisted Versioning' section in the 'Plugin Packaging and Versioning' chapter) for more details.

NotifyingResultListener no longer supports sounds

`System sounds` and `Custom sounds` has been removed from `NotifyingResultListener` in BasicSteps. `Run command` is the only type of notification supported now.

ShortNameAttribute Deleted

In place of the current usage of `ShortNameAttribute`:

```
[ShortName("myInst")]
public class MyInstrument : Instrument{

}
```

The `Name` of the instrument can instead be set in the constructor:

```
public class MyInstrument : Instrument{
    public MyInstrument(){
        Name = "myInst";
    }
}
```

PlatformInteraction Reworked/Deleted

`PlatformInteraction` has been reworked to be more flexible and extensible. Use `UserInput.Request` instead of `PlatformInteraction`. An example of this can be found in `%TAP_PATH%\Packages\SDK\Examples\PluginDevelopment\GUI\UserInputExample.cs`.

TapVersion Renamed

`TapVersion` has been cleaned up and renamed to `SemanticVersion`.

As part of the clean-up the method to get TAP version (`TapVersion.GetTapEngineVersion()`) has been deleted.

You can now do it one of these ways: - `new Installation(Directory.GetCurrentDirectory()).GetOpenTapPackage().Version` - gets the version number of the *installed* OpenTAP package. - `PluginManager.GetOpenTapAssembly().SemanticVersion` - gets the version number of the *loaded* OpenTAP assembly.

Minor changes

- `SibelingStepControlProvider` renamed to `SiblingStepControlProvider`
- `IResultStore.GetAverageDuration` return value changed from `TimeSpan` to `TimeSpan?` (nullable timespan)
- `SessionLogs.Load()` method renamed to `SessionLogs.Initialize()`
- Removed `TestPlanRun.AbortRequested` event. To see if the current thread is aborted check `TapThread.Current.AbortToken`,
- Use `System.OperationCanceledException` instead of `Keysight.Tap.AbortException`
- `TestPlan.Sleep()` has been removed. Instead use `TapThread.Sleep()`.
- `TestPlan.Abort()` and `TestPlan.AbortException` has been removed. Instead `TestPlanRun.MainThread.Abort()` can be used.
- `EnabledIfAttribute.IsEnabled` has been changed to use the new reflection system. See TypeData.

ITypeData / IMemberData

This applies if you previously have been using `object.GetType()` for C# reflection. Many APIs in OpenTAP use `TypeData.GetTypeData(object)` or `TypeData.FromType`. This system is similar to classic C# reflection, but has some features that makes it simpler and more flexible with regards to dynamic types. - Instead of `object.GetType()` use `TypeData.GetTypeData(object)`. - For member info, when previously `PropertyInfo.PropertyType` was used, now use `IMemberData.TypeDescriptor`.

Builds

The assembly Keysight.Tap.Sdk.MSBuild.dll has been renamed to Keysight.OpenTap.Sdk.MSBuild.dll. This means any C# project file has to be changed where the following is stated:

```
<UsingTask TaskName="Keysight.Tap.Sdk.MSBuild.PackageTask" AssemblyFile="$(TAP_PATH)\Keysight.Tap.Sdk.MSBuild.dll" />
```

This should be changed to

```
<UsingTask TaskName="Keysight.OpenTap.Sdk.MSBuild.PackageTask"
AssemblyFile="$(TAP_PATH)\Packages\SDK\Keysight.OpenTap.Sdk.MSBuild.dll" />
```

In previous build scripts you might have had a line similar to:

```
Keysight.Tap.Install.exe --uninstall --clean --install --branch release9x
```

In OpenTAP 9.0, you'll need to change to using the –version specifier.

```
Keysight.Tap.Install.exe --uninstall --clean --install --version 9.0
```

ICliAction

`Execute()` method now takes a `CancellationToken` as argument. -> `Execute(CancellationToken)`.

package.xml file

The content of the package.xml file has changed. For example this package.xml from TAP 8.x:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package Name="Plugin Example"
         xmlns="http://keysight.com/schemas/TAP/Package"
         InfoLink="http://www.keysight.com/"
         Configuration="MyConfiguration"
         Version="$(GitVersion)">
  <Description>Some example plugin.</Description>
  <Files>
    <File Path="SomePlugin.dll" Obfuscate="true" SetAssemblyInfo="Version,Configuration" Sign="Keysight Technologies,
Inc."></File>
```

```
    <File Path="SomeSampleData.txt"></File>
  </Files>
</Package>
```

Needs to be modified to this for OpenTAP 9.x:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package Name="Plugin Example"
         xmlns="http://opentap.io/schemas/package"
         InfoLink="http://www.keysight.com/"
         Version="$(GitVersion)">
  <Description>Some example plugin.</Description>
  <Files>
    <File Path="SomePlugin.dll">
      <SetAssemblyInfo Attributes="Version"/>
      <ObfuscateWithDotfuscator/>
      <!--ObfuscateWithObfuscar/-->
      <Sign Certificate="Keysight Technologies, Inc."/>
    </File>
    <File Path="SomeSampleData.txt"></File>
  </Files>
</Package>
```

The `ObfuscateWithDotfuscator`, `ObfuscateWithObfuscar` and `Sign` actions in the above package.xml requires installing some Keysight OpenTAP plugins (Dotfuscator, Obfuscar and Sign).

PackageManager

- `PluginInstaller` class has been removed. Instead use package cli actions.
- `IPackageIdentifier.PackageName` renamed to `Name`
- `RepositoryManager` class has been removed
- `FileRepositoryManager` renamed to `FilePackageRepository`
- `HttpRepositoryManager` renamed to `HttpPackageRepository`
- `VersionMatcher` class has been removed. Instead use `VersionSpecifier` and `SemanticVersion`
- `PackageReference` renamed to `PackageIdentifier`
- Package cli action have changed command line arguments:
  - `tap-dir` renamed to `target`
  - `cpu` renamed to `architecture`
  - `prerelease` has been removed. Instead use –version
  - Obsolete arguments have been removed
- All methods from `IPackageRepository` have changed.

Version specification

Specifying a package version now defaults to exact match and compatible is now opt in with prepending '^'. You can now also use '`Any`' to get the latest version regardless of release type. E.g.:

Specification	Command
A version compatible with 9	`^9`
Version 9.0.201	`9.0.201`
Any version	`Any`

Aborting Threads / TapThreads

In .NET Framework it was possible to call Thread.Abort, but this feature is not supported across platforms. The concept of TapThreads has been introduced to manage relationships between threads. TapThreads can be 'soft' aborted. Each has a cancellation token that can be used. TapThreads are heriarchical, so TapThreads remembers which thread spawned them. If a TapThread is aborted, so are threads it spawned during its life time. To support soft aborting test plans: - Use `TapThread.Sleep()` instead of `Thread.Sleep()` (Previously also `TestPlan.Sleep()`) - Use `TapThread.Current.AbortToken` for possibly long running operations that take a CancellationToken. For example for socket communication. - Use `TapThread.Current.AbortToken.IsCancellationRequested` to see if the current thread or one of the 'parent' threads

was canceled. - For operations that may not be aborted, `TapThread.WithNewContext` can be used to run an operation i a ‘virtual’ TapThread outside the threading heirarchy.

IPropGridControlProvider

The custom GUI component system has been changed to something more dynamic and flexible in order to support alternative user interfaces, for example CLI and REST-APIs. It is still possible to create custom WPF controls using the new IControlProvider interface inside `Keysight.OpenTap.Wpf`, see SDK examples for an example of this.

For more flexibility, if something can be modified using an already existing control, like a string or drop-down type, we suggest you create an *annotator* instead. This has the benefit that it reuses the already developed TAP controls and is supported on cross platform APIs, without any extra work. Again refer to the example code for more information.

`IControlProvider` does not have the `PropGridControlPosition` property anymore. Instead `LayoutAttribute` can be used to control the positioning of a control inside on a settings level.

Migrating TAP Plugins from Version 7.x to Version 8.x

TAP 8.x contains a number of breaking changes relative to TAP 7.x - hence the change in major version number. This page contains help on how to migrate from TAP 7.x.

TypesToSearchFor replaced by ITapPlugin

`TypesToSearchFor` has been replaced with `ITapPlugin`. This means that your plugin must inherit from the `ITapPlugin` interface to be recognized as a plugin. This also removes `AddSearchType(Type type)`.

Packages Extension Changed

`.TapPlugin` is now called `.TapPackage` and the extension `.TapPackages` is supported for projects that may bundle multiple packages in one file (it is basically a zip of `.TapPackage`'s).

PluginPackageManager renamed to PackageManager (to simplify things).

TAP PluginPackageManager has been renamed to PackageManager, this effects these projects:

Project Assembly Name

- `Tap.PluginPackageManager -> Tap.PackageManager`
- `Tap.PluginPackageManager.Gui -> Tap.PackageManager.Gui`
- `Tap.PluginPackageManager.UnitTests -> Tap.PackageManager.UnitTests`

Project Namespace

- `Keysight.Tap.PluginPackageManager -> Keysight.Tap.Package`
- `Keysight.Tap.PluginPackageManager.Gui -> Keysight.Tap.PackageManager.Gui`
- `Keysight.Tap.PluginPackageManager.UnitTests -> Keysight.Tap.Package.UnitTests`

Plugin Subdirectory Support (Optional)

You can now build and release plugins inside a subdirectory of TAP. However, some changes are needed:

1. First, change .csproj PackageTask Dir's Property to simply $(TAP_PATH).
2. Then change the output directory to $(TAP_PATH)_DIRECTORY_STRUCTURE_PLUGIN.
3. Then append all files in package.xml with your desired directory structure.

IDisposable is removed from Resource

IDisposable was not used by TAP. If you were depending on IDisposable for something else, it can be implemented on your class.

.NET Framework Changed to 4.6.2

We have changed the targeted framework for TAP to .NET 4.6.2 (required for cross platform support). TAP plugins will need to target this version of .NET as well.

To build target .NET 4.6.2 using Visual Studio 2015 or 2017, you need this developer pack .

Changes to ScpiInstrument

Removed usage of VISA-COM

This means that the `IFormattedIO488 formatter` is no longer exposed as a protected field. This change is mandated because VISA-COM is not crossplatform. Any calls that need that low-level access are available as higher-level methods in ScpiInstrument.

SRQ Handling

The `SetSRQHandler`, `RemoveSRQHandler`, `SetupSRQ`, and `WaitForSRQ` have been removed. These have been replaced with an event called `SRQ`. The reason for the change is that it reduces the size of the public interface and that the old approach was very hard to explain.

Now the SRQ handlers are called like a normal C# event whenever an SRQ happens. Multiple users can also attach SRQ's to the same instrument. SRQ handlers are automatically reattached even if the instrument is closed and reopened. This also means that if an SRQ is attached it must be detached when it's no longer needed.

PluginManager.Search replaced by PluginManager.SearchAsync

As the name suggests Searching is now Async by default. You can wait for it using `PluginManager.SearchAsync().Wait()` if you want to, but it should not be necessary, as any calls to the PluginManager that need the search to be completed will now automatically wait internally.

Calling PluginManager.SearchAsync is no longer required. It can improve performance (search in the background) if you are able to call it before you need the plugins. If it is not called, searching is performed automatically when needed (causing a delay, roughly 0.5 seconds).

ResultParameters

ResultParameters from TestStepRun now uses an IReadOnlyList, this may effect some LINQ statements.

Tap.Licensing.LicenseManager is removed

Tap.Licensing.LicenseManager is removed. If you were depending on LicenseManager for something else, it can be implemented in your class.

Obsolete functionality removed

Teststeps

- FixedCountLoop `Use Repeat step instead`
- WhileLoop `Use Repeat step instead`

Methods

- LogBufferMode `TAP only uses filestream as buffer now`
- Tap.Serializer `Use TapSerializer instead`
- Serializer.SerializeToString `Use new TapSerializer().SerializeToString`
- TestPlan.Load(Stream) `Use Load(Stream, String) instead`
- PackageManager.GetAllVersions(PackageDef) `Use GetCompatibleVersions instead`
- PackageManager.GetAllVersions(String) `Use GetCompatibleVersions instead`
- `MacroPathAttribute.Expand() Use MacroString.Expand on a MacroString instance`

Interfaces

- ITestStepParent no longer has a Name nor does it support GetEnabledTestSteps(). If the instance is an ITestStep it can be casted to that and the name can be retrieved from there. Instead of `GetEnabledTestSteps()` use `ChildTestSteps.Where(step => step.Enabled)`.

IDynamicStep

GetStepData and the argument to GetStep has been removed. Instead, it is now supported to set properties on the IDynamicStep which then gets serialized and can be used inside GetStep(). The easiest way to migrate, is to simply add a property that stores the data that previously GetStepData() returned.

For example, this (TAP 7x):

```
public class MyStep : TestStep, IDynamicStep{
    string nonSerializedField = "MyData";
    public string GetStepData(){ return "MyData";}
    public ITestStep GetStep(string data){ return new MyStep(){nonSerializedField = data};}
    public override Run(){}
}
```

Would be functionally equivivalent to this (TAP 8x):

```
public class MyStep : TestStep, IDynamicStep{
    [Browsable(false)]
    public string mySerializedData
    {
       get { return nonSerializedField; }
       set { nonSerializedField = value; }
    }

    string nonSerializedField = "MyData";
    public ITestStep GetStep(){ return new MyStep(){nonSerializedField = mySerializedData};}
    public override Run(){}
}
```

Attributes

MacroPathAttribute ([MacroPath])

This used to be a way to have macros supported in a string. Now simply use the MacroString type as a return of your property and conditionally if you want to use it as a FilePath, add the [FilePath] attribute. An important difference is that MacroString is not necessesarily a path, so it will not automatically expand to an absolute path, If you need an absolute path call `System.IO.Path.GetFullPath` with the result of `MacroString.Expand()`. Example:

```
class MyTestStep: TestStep {

  [FilePath] // A MacroString that is also a file path.
  public MacroString Filename { get; set; }

  public MyTestStep(){
    // 'this' useful for TestStep instances.
    // otherwise a MacroString can be created without constructor arguments.
    Filename = new MacroString(this) { Text = "MyDefaultPath" };
  }
  public override Run(){
     log.Info("The full path was '{0}'.", Path.GetFullPath(Filename.Expand()));
  }
}
```

Properties

- TestPlan.Name is now read-only and no longer settable. Remove usages of setting the name. The name of the file without the extension is the name of the TestPlan.
- With newly added support for Switch Matrix, ViaPoint (Connection.Via) has to be explicitly cast as either ‘SwitchPosition’ or ‘SwitchMatrixRow/SwitchMatrixColumn’. Otherwise you’ll receive an ‘Unable to convert from’ error.

Tap.Package build tasks

The `Keysight.Tap.Package.exe` executable which houses the MSBuild tasks for building the plugins has been renamed to `Keysight.Tap.Sdk.MSBuild.dll`. This to simplify things and get rid of the Tap.Package.exe (so we only have Tap.PackageManager.exe).

This requires that plugins that used the templates of TAP 7.x or earlier needs to be modified. The plugins need the .csproj file modified.

A legacy plugin .csproj file will look like this:

```
  <UsingTask TaskName="Keysight.Tap.Package.PackageTask" AssemblyFile="$(TAP_PATH)\Keysight.Tap.Package.exe" />
  <UsingTask TaskName="Keysight.Tap.Package.SetAssemblyInfoTask" AssemblyFile="$(TAP_PATH)\Keysight.Tap.Package.exe"
/>
  <Target Name="BeforeBuild" Condition="'$(Configuration)' == 'Release'">
    <SetAssemblyInfoTask FilePath="Properties\AssemblyInfo.cs" AssemblyInformationalVersionFromGit="True" />
  </Target>
  <Target Name="AfterBuild" Condition="'$(Configuration)' == 'Release'">
    <GetAssemblyIdentity AssemblyFiles="$(TargetPath)">
      <Output TaskParameter="Assemblies" ItemName="TargetInfo" />
    </GetAssemblyIdentity>
    <PackageTask Dir="$(TAP_PATH)" ConfFile="$(ProjectDir)\package.xml" />
  </Target>
```

This needs to be modified to look like this:

```
  <UsingTask TaskName="Keysight.Tap.Sdk.MSBuild.PackageTask" AssemblyFile="$(TAP_PATH)\Keysight.Tap.Sdk.MSBuild.dll"
/>
  <Target Name="AfterBuild" Condition="'$(Configuration)' == 'Release'">
    <GetAssemblyIdentity AssemblyFiles="$(TargetPath)">
      <Output TaskParameter="Assemblies" ItemName="TargetInfo" />
    </GetAssemblyIdentity>
    <PackageTask Dir="$(TAP_PATH)" ConfFile="$(ProjectDir)\package.xml" />
  </Target>
```

The `SetAssemblyInfoTask` used previously was responsible for updating the version information of the individual files in a plugin. If this is important you need to modify the `package.xml`. Here's an example that would do the same as previously:

```
<File Path="MyPlugin\Tap.Plugins.MyPlugin.dll" Obfuscate="true" SetAssemblyInfo="Version,Configuration" />
```

Migrating TAP Plugins from Version 6.x to Version 7.x

License N7400 is no longer supported, go generate a new temp license using the TAP homepage

ResultSource.Publish Rename

ResultSource.Publish() does no longer contain a overload for storing arrays of arrays. You have to use ResultSource.PublishTable() instead.

If you did something like the sample below you would need to call `PublishTable` instead of `Publish`.

```
double[] array1 = new double[10];
double[] array2 = new double[10];

// .. populate the elements of array1 and array2

// TAP 6.x
Results.Publish("My Result", new List<string> { "Column A", "Column B" }, array1, array2);
// TAP 7.x
Results.PublishTable("My Result", new List<string> { "Column A", "Column B" }, array1, array2);
```

DisplayName and **Description** attributes no longer supported

These were obsoleted in 6.0, see here .

New examples can be found here .

Temp licenses (N7400) generated by the TAP homepage no longer supported

A new link is now available on the homepage to generate KS8400A licenses supported by TAP 7.x.

Migrating TAP Plugins from Version 5.x to Version 6.x

Only minor changes have been introduced between TAP 5.x and 6.x but enough to justify a jump in the major version number.

Uninstall old versions of TAP and delete the old TAP program directory

Namespace Rename

Keysight.Tap.TapPlugin now renamed to **Keysight.Tap.Plugins**

It is recommended that your KS plugins are all placed within this namespace. Note the SDK templates do not include the company name Keysight, this is up to the end customer to define.

Plugins Rename

Keysight.Tap.Plugins now renamed to **Keysight.Tap.PluginManager**

PlatformSettings Rename

Keysight.Tap.PlatformSettings now renamed to **Keysight.Tap.EngineSettings**

“Fixed count loop” and “While loop” no longer available (still supported but obsolete).

Instead we have a **‘Repeat’** step that offer the same functionality as the other two but in a simpler and more intuitive way.

DisplayName, ShortName and Description attributes now obsolete

They are all replaced by the **Display** attribute that offer all the above functionality. We know this is a big change but we are doing it as the current display attribute is not going to be supported on cross platform (.NET core). another benefit of using the new Display attribute is that it offers a way to ‘order’ the properties in the grid (no need to use lots of ‘spaces’ or ’ to change the order.

Database schema changed in TAP 6.0

TAP will automatically update your database schema to the new version. Prior to doing so a copy will be made ensuring that you will still have your old data/format available.

Migrating TAP Plugins from Version 4.x to Version 5.x

TAP 5.x contains a number of breaking changes relative to TAP 4.x - Hence the change in major version number. This page contains help on how to migrate from TAP 4.x

- **VS2015** If you use VS2015 please ensure that you have Update 1 installed (the first version has some compiler bugs)

Namespace Rename

In order to comply with Keysight's Engineering Standards, "Keysight" has been added to **all** namespace in the Tap Core project. For instance, this results in `Tap.Engine.TestModule` becoming `Keysight.Tap.Engine.TestModule`.

This also influences dll references. E.g. `Tap.Engine.dll` is renamed to `Keysight.Tap.Engine.dll`

Resource Log Rename

Resource.log has been renamed to Resource.Log. Previously there was a public 'log' and a protected 'Log', these have been consolidated. Affects Instrument, Dut, and ResultListener classes.

Tap TraceSource

Tap.TraceSource now uses a new enumeration of possible log types to better reflect typical uses.

This means that anywhere Tap.TraceSource is used, the following must be replaced:
`TraceEventType` or `System.Diagnostics.TraceEventType` must be replaced by `LogEventType`.

Futermore the following log types must be replaced:
* `TraceEventType.Start` and `TraceEventType.Stop` must be replaced by `LogEventType.info`. * `TraceEventType.Verbose` must be replaced by `LogEventType.Debug`.

Removal of Ksf.dll

TAP 5.x will no longer use the Ksf.dll introduced in TAP 4.x.

Any references to this DLL should be removed from plugin projects.

Change in Result storage and ResultListeners.

IVector is no longer used as the internal datatype to handle results in TAP. Instead it now uses a simplified ResultTable class. This is the type that a ResultListener will receive when using the OnResultPublished method which was named AddResult previously. Moreover, the OnResultPublished method takes a Guid in its parameters instead of a TestStepRun

The previous methods and properties related to IVector can be mapped as follows to the ResultTable: 1. Vector.Channels —> ResultTable.Colums 2. Vector.Points —> ResultTable.Rows

For the TestStep side the ResultProxy now has 4 methods of results entry, and TestStepResult has been removed: 1. PublishResult(T result)
2. PublishResult(string name, T result)
3. PublishResult(string name, List columnNames, params IConvertible[] results)
4. PublishResult(string name, List columnNames, params Array[] results) **FASTEST**

Number 1 and 2 take an arbitrary object and uses reflection to retrieve public properties. This is similar to how the TAP Gui presents teststep properties in the step settings panel.
Number 3 is identical in functionality to how the TAP 4.x StoreResult worked when given a

TestStepResult.
Number 4 is similar to method 3, but instead takes a number of arrays. This is useful if there is a need to enter a large amount of data.

This example shows how the RatingStep class from the Demonstration Plugin is modified to use PublishResult instead of TestStepResult. Initially, this is the code we start with:

```
public override void Run()
{
    double totalTime = ChargeTime.Value + DischargeTime.Value;
    if (totalTime > RatingBLimit)
    {
        CalculatedRating = Rating.C;
    }
    else if (totalTime > RatingALimit)
    {
        CalculatedRating = Rating.B;
    }
    else
    {
        CalculatedRating = Rating.A;
    }

    var resulttype = new TestStepResultType() { Name = "Rating", DimensionTitles = new List<string> { "Charge Time"
    Results.StoreResult(resulttype, ChargeTime.Value, DischargeTime.Value, totalTime, CalculatedRating.ToString());
}
```

image

Here is a single line replacement: Note that we no longer need to construct a TestStepResult.

```
Results.Publish("RatingResult", new
    List<string> {"ChargeTime", "DischargeTime", "TotalTime", "Rating"},
    new IConvertible[] { ChargeTime.Value, DischargeTime.Value, totalTime, CalculatedRating});
```

image

Additionally, one could use a new feature of defining a result class, and then constructing and publishing that class. An example of the class definition and Publish statement are shown below.

```
[DisplayName("Rating")]
1 reference | BillBush, 7 days ago | 1 author, 2 changes
public class RatingResult
{
    [DisplayName("Charge Time")]
    1 reference | BillBush, 7 days ago | 1 author, 1 change
    public double ChargeTime { get; set; }
    [DisplayName("Discharge Time")]
    1 reference | BillBush, 7 days ago | 1 author, 1 change
    public double DischargeTime { get; set; }
    [DisplayName("Total Time")]
    1 reference | BillBush, 7 days ago | 1 author, 1 change
    public double TotalTime { get; set; }
    [DisplayName("Rating")]
    1 reference | BillBush, 7 days ago | 1 author, 2 changes
    public Rating Rating { get; set; }
}
```

```
Results.Publish(new RatingResult
    {
        ChargeTime = ChargeTime.Value,
        DischargeTime = DischargeTime.Value,
        TotalTime = totalTime,
        Rating = CalculatedRating
    });
```

This excerpt from the developers guide discusses using the new PublishResult calls.
How_to_publish_results_from_a_TestStep.docx

There are several new examples that deal with results.

For example code on publishing results, see the ...Examples.Plugin.cs file.

For example code on handling the new OnResultPublished calls to the ResultListeners, see the ...TAPExamples.Plugin.cs file

`ShortName` is removed from the Resource class (and thus also from the Instrument class)

Using the `ShortName` PROPERTY from the Resource class in Release4x should throw a deprecation warning. In 5x, this property will be completely removed from the Resource class and must accordingly be removed from plugins relying on this property.

`ShortName` must be set through an ATTRIBUTE to the related class as shown by the following example from the PowerAnalyzer class found in the Demonstration plugin:

```
namespace TapPlugin.BatteryDemo
{
    [DisplayName("Power Analyzer")]
    [Description("Insert a description here")]
    [ShortName("PSU")]
    public class PowerAnalyzer : Instrument
    {
        #region Settings
        public double CellSizeFactor { get; set; }
        #endregion
...
```

Tap.Server.Wcf changes

The `RemoteWcfClient` class has been renamed to `WcfClient` and the method to create a new Wcf-Server process has been extracted into a new class named `WcfServerProvider`. This method had a parameter option called `isHidden`, this is instead a property named `ShowDebugConsole`. Below is an example of the new usage: (There is also a new example in the SDK examples, typically installed at ..Examples.Api.Api.csproj)

```
 class Program
{
    static void Main(string[] args)
    {
        // Create the WCF server
        WcfServerProvider server = new WcfServerProvider() { Port = 1330, ShowDebugConsole = true };
        server.PluginSearchPaths = new List<string> { @"C:\MyTapPath" };
        server.StartLocalServerProcess();

        Thread.Sleep(500);

        // Create the WCF client
        WcfClient client = new WcfClient() { Address = "localhost", Port = 1330 };
        client.Open();
        client.LoadTestPlan(new FileStream(@"C:\MyTapPath\MyPlan.TapPlan", FileMode.Open));
        client.RunTestPlan();

        Thread.Sleep(1500);

        client.AbortRun();

        Thread.Sleep(1500);

        client.Close();

        server.Shutdown();
    }
}
```

ComPort Type

If your plugin requires the use of ComPort Type, add a reference to Keysight.Tap.Plugin.RemoteTestBench.dll and a using directive to

Keysight.Tap.Plugin.RemoteTestBench in the file where it is being used.

General information to upgrade plugins: Changes to Solution, Projects and Namespaces

General

Solutions, Projects, Output dlls and project directories should have consistent names as per the following naming standards: - Namespaces will be of the form Keysight.Tap.Plugins.[Name]. For specific cases such as ResultListeners the Namespaces will follow Keysight.Tap.Plugins.[Type].[Name]. For example, Keysight.Tap.Plugins.ResultListener.Excel - Output Dlls will be of the form Keysight.Tap.Plugins.[Name].dll. For specific cases such as ResultListeners the dll names will follow Keysight.Tap.Plugins.[Type].[Name].dll. For example, Keysight.Tap.Plugins.ResultListener.Excel.dll

Instructions

- Rename the Solution and Projects as per Keysight Naming Standards. This will depend on the plugin type. For example, ResultListeners would be named Keysight.Tap.Plugins.ResultListener.[YourPluginName]
- Right-click the Solution, select Rename, and enter the new name
- Right-click the project in Solution Explorer, select Rename, and enter the new name
- Open Package.xml and update ‘File Path’ attribute
- In the project directory, update .gitlab-ci.yml. Change release[N]x to release5x
- Update references:
- Unload project, right click and edit: RootNamespace, AssemblyName, Reference, HintPath, UsingTask etc
- Remove reference KSF.dll
- Update AssemblyInfo.cs.
- Possible modifications are Assembly version, Assembly Product
- Close Solution and update folder structure. Rename folders as required.
- Recommended structure is to have the solution file on the outermost directory with each project in a directory that matches the project name. For example:

```
Folder PATH listing
C:.
|   .gitattributes
|   .gitignore
|   .gitlab-ci.yml
|   Keysight.Tap.Plugins.ResultListener.Excel.sln
|   tree.txt
|
\---Keysight.Tap.Plugins.ResultListener.Excel
    |   ExcelListener.cs
    |   Keysight.Tap.Plugins.ResultListener.Excel.csproj
    |   package.xml
    |
    +---obj
    |   \---Debug
    |       |
    |       \---TempPE
    \---Properties
            AssemblyInfo.cs
```

- Open the *.sln file in a text editor and update the project path. `Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "Keysight.Tap.Plugin.ResultListener.Excel", "Keysight.Tap.Plugin.ResultListener.Excel\Keysight.Tap.Plugin.ResultListener.Excel.csproj", "{9124CCEA-D3ED-4016-8F2F-4B6260405D30}"`
- Reopen the solution in Visual studio and ensure it loads correctly
- Update namespaces, references, inheriting classes and interfaces
- You can now manually update namespaces and using statements [recommended]
- Alternatively, if you have ReSharper installed, you can use the [Right click on project]>>Refactor>>Adjust Namespaces feature
- Follow the migration guide listed above to make appropriate code changes for migration
- Build and test your code.

Updating source control

Note: It is also possible to make the changes below using Git within Visual Studio, without command line.

- Open Git Command line and close Visual Studio
- On executing git status, you will notice that git still points to the older filenames and marks them as deleted:

```
On branch 2_4xTo5xUpgrade
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitlab-ci.yml
        deleted:    ResultListener.Excel.sln
        deleted:    TapPlugin.ResultListener.Excel/ExcelListener.cs
        deleted:    TapPlugin.ResultListener.Excel/Properties/AssemblyInfo.cs
        deleted:    TapPlugin.ResultListener.Excel/TapPlugin.ResultListener.Excel.csproj
        deleted:    TapPlugin.ResultListener.Excel/package.xml
```

- Delete the previous solution and the older directory reference using git rm -r

```
E.g.
git rm ResultListener.Excel.sln
git rm -r TapPlugin.ResultListener.Excel
```

- Add the new solution file and the project folder using git add

```
E.g
git add Keysight.Tap.Plugins.ResultListener.Excel.sln
git add  Keysight.Tap.Plugins.ResultListener.Excel/
```

- verify your changes using git status

```
On branch 2_4xTo5xUpgrade
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .gitlab-ci.yml
        renamed:    ResultListener.Excel.sln -> Keysight.Tap.Plugin.ResultListener.Excel.sln
        renamed:    TapPlugin.ResultListener.Excel/ExcelListener.cs ->
Keysight.Tap.Plugin.ResultListener.Excel/ExcelListener.cs
        renamed:    TapPlugin.ResultListener.Excel/TapPlugin.ResultListener.Excel.csproj ->
Keysight.Tap.Plugin.ResultListener.Excel/Keysight.Tap.Plugin.ResultL
istener.Excel.csproj
        renamed:    TapPlugin.ResultListener.Excel/Properties/AssemblyInfo.cs ->
Keysight.Tap.Plugin.ResultListener.Excel/Properties/AssemblyInfo.cs
        renamed:    TapPlugin.ResultListener.Excel/package.xml -> Keysight.Tap.Plugin.ResultListener.Excel/package.xml
```

- You can now go back to Visual Studio and commit your changes. Alternatively, you can also commit from git command line

Small API Changes

HandlesTypesAttribute reordering of arguments. Now must put Priority as the first argument and without specifying the parameter by name. Eg: [HandlesTypes(typeof(ComPort), Priority = 16)] becomes [HandlesTypes(16, typeof(ComPort))]

PluginManager becomes Plugins

TapSerializer.Serialize becomes Serializer.Serialize or Serializer.SerializeToString depending on what you want. TapSerializer.Deserialize becomes Serializer.Deserialize

Ksf.Diagnostic.Event becomes Keysight.Tap.Diagnostic.Event

Tap.BenchSettingsAttribute no longer exists.

RemoteTestBenchSettings.GetCurrent() becomes RemoteTestBenchSettings.Current.

[MacroPath(AllowDateTime = true)] becomes [MacroPath]

`ComponentSettings.SettingsDirectory = settingsSetDir;` changes to `ComponentSettings.SetSettingsProfile("Bench", dir);`

`log.TraceInformation("Settings: " + ComponentSettings.SettingsDirectory);` changes to `log.TraceInformation("Settings: " + ComponentSettings.GetSettingsDirectory("Bench"));`

`SessionLogs.Rename(PlatformSettings.Current.SessionLogPath.Expand(DateTime.Now));` changes to `SessionLogs.Rename(MacroPathAttribute.Expand(PlatformSettings.Current, "SessionLogPath", date: DateTime.Now));`

Migrating TAP Plugins from Version 3.x to Version 4.x

This page is now maintained at
https://wiki2.collaboration.is.keysight.com/display/testAutomationPlatform/3.x+to+4.x

TAP 4.x contains a number of breaking changes relative to TAP 3.x - Hence the change in major version number. This page contains help on how to migrate from TAP 3.x

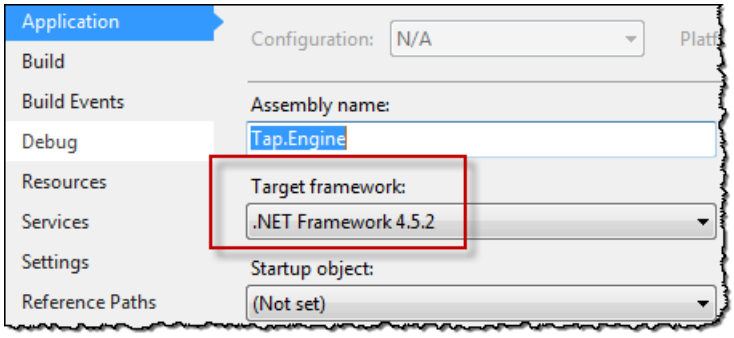
(If you are currently using 2.x then follow this guide: **Migrating from TAP 2.x to 3.x**)

Ksf.dll

Tap executables now reference the Ksf.dll library. You should modify your TAP related project to reference this assembly. The Ksf dll can be found in the TAP install folder.

.NET version has been changed to 4.5.2

TAP plugins compiled for previous versions of .NET will no longer work with TAP. The plugin needs to be updated to .NET 4.5.2 and recompiled. That change is made on the Project Properties screen, as shown here

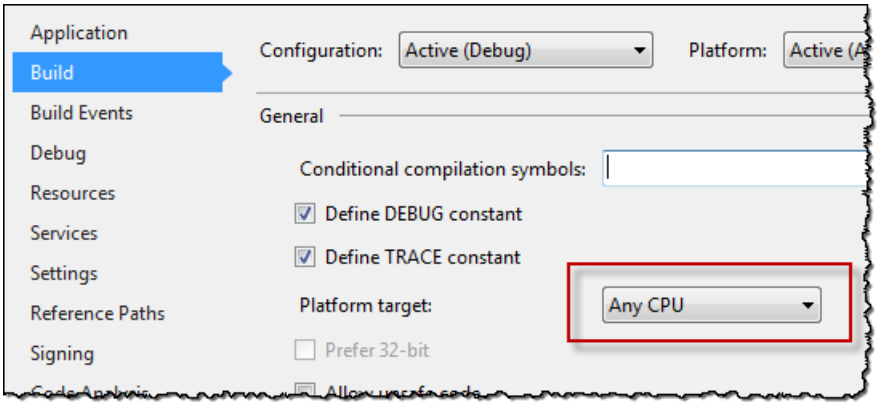


image

TAP 64-bit

TAP now supports 32 bit (x86) and 64 bit (x64) versions and installers. You should pick up the appropriate one from the Tap home page.
The 32 bit version will typically install to C:Files (x86) The 64 bit version will typically install to C:Files

Plugin developers should probably set your platform target to be Any CPU, as specified on the Project/Build page, as shown here.



image

AvailableValuesAttributes moved to Tap.Engine, and changed namespace from

Tap.Gui.Controls to Tap

The AvailableValuesAttribute has been moved so a recompile of plugins might be necessesary.

MacroPathAttribute

For FileName macros. The system has been changed from a class based one to an attribute based one, so now its possible to write the following (example from Tap.Engine project):

```
/// <summary>
/// Where the session logs are saved. Must be a valid path.
/// </summary>
[DisplayName("General \\ Log Path")]
[Description("Where to save the session log file. This setting only takes effect after restart.")]
[MacroPath(StaticOnly = true)]
public string SessionLogPath { get; set; }
```

In places where MacroPath.Expand was used beforem the MacroPathAttribute.Expand method can be used to expand macros.

IPropGridControlProvider refactored

Custom controls can now use MemberInfo instead of PropertyInfo. A method can be exposed as a button using `[Browsable(false)]`. To fix issues, simply change IPropGridControlProvider implementations to use MemberInfo instead of PropertyInfo. You will also need to change CreateContentControl to simply CreateContent.

DataResult and Optimized Trace Source

The DataResult class and Trace Source from KSF is now used instead or in addition to the original version. To fix issues make sure to add a reference to the library Ksf.dll in the TAP folder.

Tap.TraceSource

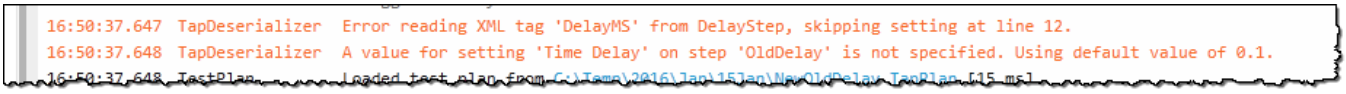
For improved performance we have moved away from System.Diagnostics.TraceSource and to a custom Tap.TraceSource implementation (Core component in KSF). This means that anyhere System.Diagnostics is imported or TraceSource is used, replace usages of TraceSource with Tap.TraceSource.

Visual Studio 2015

All the Tap projects have been moved to Visual Studio 2015.

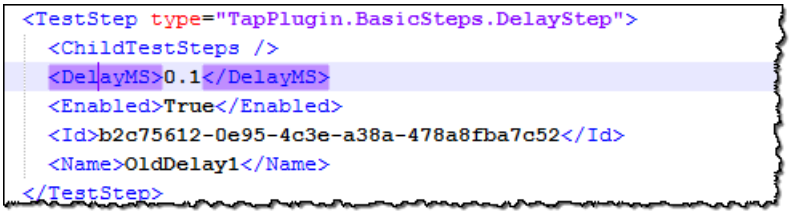
Change in DelayStep test step

The DelayStep teststep has been modified so that the (incorrectly named) DelayMS property is longer supported, and has been replaced with a new DelaySecs property. Users with testplans that used the DelayStep property will see this error on loading of their testplan.



image

User could do a mass update of testplan files by changing DelayMS to DelaySecs. For example, you would change these values:



image

ResultListeners

The ResultListener.AddResult has changed signature to use the more efficient KSF IVector datastructure. This causes the following code

```
public override void AddResult(Tap.TestStepRun run, TestStepResult result) //...
```

To now become

```
public override void AddResult(Tap.TestStepRun run, IVector results) //...
```

Also remember to add a reference to Ksf.dll from the TAP folder and add

```
using Ksf;
```

to the usings in the top of the file.

Migrating TAP Plugins from Version 2.x to Version 3.x

TAP 3.x contains a number of breaking changes relative to TAP 2.x - Hence the change in major version number. This page contains help on how to migrate from TAP 2.x

- **VS2015** If you use VS2015 please ensure that you have Update 1 installed (the first version has some compiler bugs)

TAP Verdict

Previously the verdict used by TestStep's was of type `Tap.TestStep.VerdictType`. This has been renamed to `Tap.Verdict`. Related, the verdicts `Done` and `Running` have been replaced by `NotSet`.

TestStepList.Allow... Attributes

The attributes `TestStepList.AllowAnyChild`, `TestStepList.AllowAsChildIn` and `TestStepList.AllowChildType` has been moved out of the `TestStepList` class. To fix issues related to this just erase "TestStepList.".

TraceBar.AllPassed

`TraceBar.AllPassed` boolean has been removed in favor of a `TraceBar.CombinedVerdict` property. To get same behavior as before do `TraceBar.CombinedVerdict == Verdict.Pass`.

ResultListener.OnTestStepRunCompleted/OnTestPlanRunCompleted

The parameterlist of `OnTestStepRunCompleted` no longer includes duration. Instead duration is included in TestStepRun and TestPlanRun which are arguments to the method.

ResultListener.DeleteResults removed

To simplify the ResultListener API, the `DeleteResults` method has been removed.

ResultProxy Transactions

The concept of transactions has been removed from `ResultProxy`. This means that once results has been committed, it is no longer possible to delete them.

TestPlan Nested Types

The types `TestPlan.TestStepRun` and `TestPlan.TestPlanRun` has been moved out of the TestPlan class. So now they are just called `TestStepRun` and `TestPlanRun`. ### `TestStepRun.Step` removed Instead `GetStep(TestStep.Run.TestStepId)` can now be used to get the TestStep instead.

`TestStep.PostPlanRun` Call Order

The order of TestStep.PostPlanRun is now called in reverse order of PrePlanRun. If one has two steps {A,B} the members will be called in the following order: * `A.PrePlanRun();` * `B.PrePlanRun();` * `A.Run();` * `B.Run();` * `B.PostPlanRun();` * `A.PostPlanRun();` In the previous version the order of A and B PostPlanRun would be reversed.

TestPlan.ExecuteAsync

`TestPlan.ExecuteAsync` has been removed to simplify the API. The method `TestPlan.Execute` is blocking, but can be freely invoked in a new Thread or Task.

Tap.Package SetAssemblyInfoTask Build Event

A typo has been mixed in Tap.Package, causing that all plugin csproj files containing an `SetAsemblyInfoTask`, needs to change it to `SetAssemblyInfoTask`. Note that the misspelled `SetAsemblyInfoTask`

occurs two times inside the csproj files. The result should look like on the below image. Notice the highlighted areas where it is fixed.

```
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
  <UsingTask TaskName="Tap.Package.PackageTask" AssemblyFile="$(TAP_PATH)\Tap.Package.exe" />
  <UsingTask TaskName="Tap.Package.SetAssemblyInfoTask" AssemblyFile="$(TAP_PATH)\Tap.Package.exe" />
  <Target Name="BeforeBuild">
    <SetAssemblyInfoTask FilePath="Properties\AssemblyInfo.cs" AssemblyInformationalVersionFromGit="True" />
  </Target>
  <Target Name="AfterBuild" Condition="'$(Configuration)' == 'Release'">
    <GetAssemblyIdentity AssemblyFiles="$(TargetPath)">
      <Output TaskParameter="Assemblies" ItemName="TargetInfo" />
    </GetAssemblyIdentity>
    <PackageTask Dir="$(TargetDir)" ConfFile="$(ProjectDir)\package.xml" />
  </Target>
</Project>
```

AssemblyInfoTask

Conversion Utils

The Conversion utils for converting between bands / channels / frequencies for various RF technologies has been moved from Tap.Engine to the Tap.XSignalAnalyzer plugin. The namespace has been removed so to use them one has to include TapPlugin.XSignalAnalyzer and add `using TapPlugin.XSignalAnalyzer;`.

ScpiInstrument.State Removed

The ScpiInstrument.State has been removed in favor of the `Resource.IsConnected` bool. State (`ConnectionState`) Has been moved to XSignalAnalyzer.XsaCore where it should work as before.

SystemLogs

SystemLogs has been renamed to SessionLogs. `SystemLogs.RenameTempFile` has been renamed to `SessionLogs.Rename SystemLogs.LoadTemp` has been renamed to `SessionLogs.Load`

Predefined TestStepResultTypes removed.

The predefined types now have to be defined by the plugin. For easy migration, here are the definitions:

```
public static TestStepResultType Ber = new TestStepResultType { Name = "BER", DimensionTitles = new List<string> {
"Channel", "BER [%]" } };
public static TestStepResultType Rssi = new TestStepResultType { Name = "RSSI Error", DimensionTitles = new
List<string> { "Channel", "RSSI Error [dB]" } };
public static TestStepResultType Evm = new TestStepResultType { Name = "EVM (RMS)", DimensionTitles = new
List<string> { "Channel", "EVM [%]" } };
public static TestStepResultType TxPowerError = new TestStepResultType { Name = "Tx Power Error", DimensionTitles =
new List<string> { "Channel", "Power Error [dB]" } };
public static TestStepResultType TxPower = new TestStepResultType { Name = "Tx Power", DimensionTitles = new
List<string> { "Channel", "Power [dBm]" } };
public static TestStepResultType ExpectedTxPower = new TestStepResultType { Name = "Expected Tx Power",
DimensionTitles = new List<string> { "Expected Power [dBm]", "Measured Power [dBm]" } };
public static TestStepResultType SwitchingSpectrum = new TestStepResultType { Name = "Switching Spectrum",
DimensionTitles = new List<string> { "Offset [Hz]", "Power [dBm]" } };
public static TestStepResultType RscpResult = new TestStepResultType { Name = "RSCP", DimensionTitles = new
List<string> { "Channel", "Rscp [dB]" } };
public static TestStepResultType Acp = new TestStepResultType { Name = "ACP", DimensionTitles = new List<string> {
"Channel Offset", "Power [dBc]" } };
```

Misc

- `ComPort.GetLocalComPortNames` is not static anymore. An instance of ComPort can be used to get this value instead.
- TestStepRun/TestPlanRun.Children has been removed. Use a ResultListener to get this information if needed.