Rich Client UEM ADK

We are moving to OpenKit for Real User Monitoring

DEPRECATION NOTICE: MOVING TO OPENKIT FOR REAL USER MONITORING

OpenKit for Real User Monitoring is the replacement solution which allows you to get Real User Insight for additional Digital Touch points like Java/.NET Rich Clients, PoS-Systems, ATMs,... Currently we are running an early access program covering Java and .NET based end-point implementations. If you are interested please contact christian.schwarzbauer@dynatrace.com, domink.punz@dynatrace.com or klaus.enzenhofer@dynatrace.com. We will help you getting started.

Thanks for all your feedback on this project!

Description

This is an **experimental prototype** bringing UEM features to your Java/.NET Rich Client Applications. It allows tracing activites in WinForms/WPF/SWT/AWT Clients in a similar way as Native Mobile Applications

This prototype uses the same protocol as the Mobile ADK. It requires the following steps to work
a) You need to change the source code of your rich client application to call the ADK functions
b) You need User Experience Management either on a Java Application or Web Server Enabled
c) You need a UEM License as these Rich Clients will be identified as Visits that create Page Actions.

Overview

This is an experimental release. Use at your own risk. Do not use in production.
Please let us know what you think in the forum.

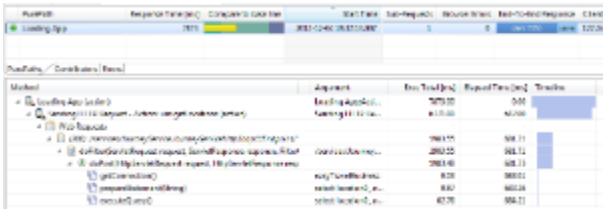
Name	Rich Client UEM ADK
Product Version	dynaTrace 5.0, 5.5, 6.0, 6.1
Author	Dynatrace Center of Excellence
Plugin Version	5.0, 5.5+
Download	.NET Downloads Sample Session with .NET Rich Client UEM Visit .NET Rich Client ADK VS Solution (Build 1.0.14.1120 from Nov 20 2014) (latest .NET Version) Sample VS Solution for 5.5+ (Build 1.0.10.0310 from Oct 03 2014) Sample VS Solution for 5.5+ (Build 1.0.0.0 from Sept 11 2014) Sample VS Solution for 5.0 Sample AWT Application including Java AWT ADK Library
License	dynaTrace Experimental Software License
Support	**DEPRECATED**

Sample for .NET

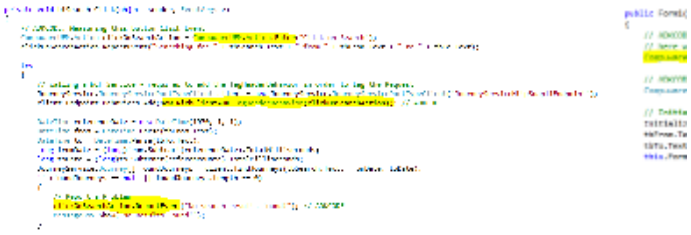
Samples and Step-by-Step Guide for .NET

Download a sample easyTravel WinForms Application. It will connect to easyTravel Business Backend Web Services running on localhost. It uses the RichClientADK to connect to a Web Server Agent on http://localhost:8079. Best is to launch one of the easyTravel UEM Scenarios.

Here are some images that show the Sample Application, Code Snippets and captured PurePaths:

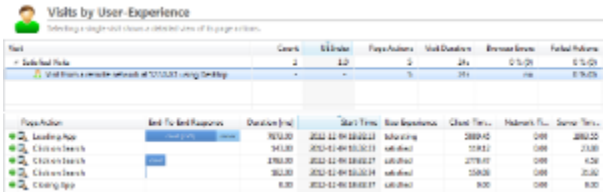


Page Action PurePath showing full End-To-End Trace from Rich Client App to Database



Code Snippet showing Page Actions that will also be linked with Web Service calls

Code



Rich Client Visit in Visits Dashlet

Upon close of the application a message box will show the captured log output of the ADK. Remember - this is a sample application 🙂

Using the ADK on your Application

Have a look at the sample project. Download

- Lasted Build 1.0.14.1120 - Full Visual Studio Solution for dT 5.x, 6.x incl releasenotes.txt
- full VS2010 Solution for dynaTrace 5.
  The project "RichClientADK" is the actual ADK. The project "easyTravelUI" is the sample WinForms Application. Explore Form1.cs to see how to use the ADK.

Step 1: Initialize ADK

Before using the ADK it needs to be initialized. The Startup method will launch a background worker thread that will later be used to send data to the configured monitorSignalURL. This call MUST BE made before making any other calls to the ADK

Here is a sample call to the Startup method:

```
// dT 5.5+ (starting Build 1.0.14.1003)
// --------------
RichClientAgent.Startup("http://localhost:8079/dynaTraceMonitor", ".NET Client",
"Desktop", "Dell", "EN", "LAN", false, true);

// dT 5.5
// --------------
RichClientAgent.Startup("http://localhost:8079/dynaTraceMonitor", ".NET Client",
"Desktop", "Dell", "EN", "LAN", false);

// dT 5.0
// --------------
CompuwareUEM.Startup("http://localhost:8079/dynaTraceMonitor", ".NET Client",
"Desktop", "Dell", "EN", "LAN");
```

void Startup(string monitorSignalURL, string applicationId, string deviceType, string manifacturer, string userlanguage, string connectionType)
Description: Will initialize the ADK

Parameter	Comment
monitorSignalURL	Configured UEM Signal URL, e.g: http://yourwebserver/dynaTraceMonitor
applicationId	The Application Name used for the captured Page Actions
deviceType	Will show up with the visit, e.g: Laptop, Desktop, ... (can be empty string)
manifacturer	Allows you to specify manifacturer name of device, e.g: DELL, Lenovo, ... (can be empty string)
userlanguage	Language setting of the user, e.g: EN, DE, FR, ... (can be empty string)
connectionType	Network Connectivity Information, e.g: WiFi, LAN, ... (can be empty string)
useGET	new for 6.1: please use TRUE true=use HTTP GET to send signal. false=use HTTP POST.

Step 2: Create Actions

A new action will start a new Page Action which will measure the time between creating the action and calling the method Leave() on it.

```
// --- dT 5.5+
// --------------
UemAction loadAppAction = UemAction.EnterAction("Loading App");
// do work
loadAppAction.LeaveAction()

// --- dT 5.0 ---
// --------------
CompuwareUEM.Action loadAppAction = CompuwareUEM.Action.Enter("Loading App");
// ... do work
loadAppAction.Leave();
```

Action Enter(string name)
Description: Will start a new Page Action.

Parameter	Comment
name	Name of the Page Action
return	Action object to be used for further logging

void Leave()
Description: Will end the Page Action

It is also possible to have child actions. Simply call EnterChildAction on an Action object.

```
// dT 5.5+
// --------------
UemAction rootAction = UemAction.EnterAction("Sample Action");
  UemAction childAction2 = rootAction.EnterSubAction("Sample Action 2");
  childAction2.LeaveAction();
rootAction.LeaveAction();


// dT 5.0
// --------------
CompuwareUEM.Action rootAction = CompuwareUEM.Action.Enter("Sample Action");
  CompuwareUEM.Action childAction2 = rootAction.EnterChildAction("Sample Action 2");
  childAction2.Leave();
rootAction.Leave();
```

Step 3: Tag Web Service Calls

When an application makes Web Service calls or regular HTTP Web Requests they can be tagged so that the captured server-side PurePaths will be linked to the Page Action.

```
// To tag a Web Service call through a Proxy class we simply add a behavior
JourneyService.JourneyServicePortTypeClient client = new
JourneyService.JourneyServicePortTypeClient("JourneyServiceHttpSoap11Endpoint");
client.Endpoint.Behaviors.Add(new RichClientADK.TagHeaderBehavior(loadAppAction));
client.getLocations();

// To tag an Http call we simply call the TagWebRequest method
HttpWebRequest request =
(HttpWebRequest)HttpWebRequest.Create("http://localhost:8079/services/JourneyService
/findJourneys");
RichClientADK.CompuwareUEM.TagWebRequest(request, null);
HttpWebResponse response = (HttpWebResponse)request.GetResponse();
```

void RichClientADK.TagHeaderBehavior(Action parentAction)
Description: Will make sure to tag the outgoing HTTP Request of the Web Service call

Parameter	Comment
parentAction	Server-Side PurePath will be linked to that parent action. If NULL the Web Request will be linked to the current open Action

void TagWebRequest(HttpWebRequest request, Action parentAction)
Description: Will tag this request by adding an HTTP Header

Parameter	Comment
request	HttpWebRequest object
parentAction	Server-Side PurePath will be linked to that parent action. If NULL the Web Request will be linked to the current open Action

Step 4: Log context information

Additonal to monitoring actions it is possible to log additional context information on page actions. In the current version of the ADK we allow to report a string value.

```
clickOnSearchAction.ReportEvent("No search results found!");
```

void ReportEvent(string name)
Description: Logs a string value on the Page Action. Can be seen in Page Action PurePath

Parameter	Comment
name	Any string value that should be logged

Step 5: Shutdown ADK

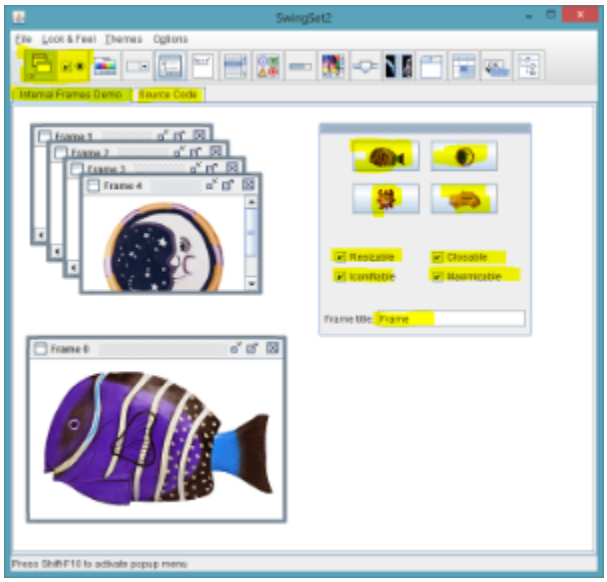
Before the application stops it is required to shutdown the ADK. This allows the ADK implementation to send any unsent monitoring data and also gracefully stop the background worker thread

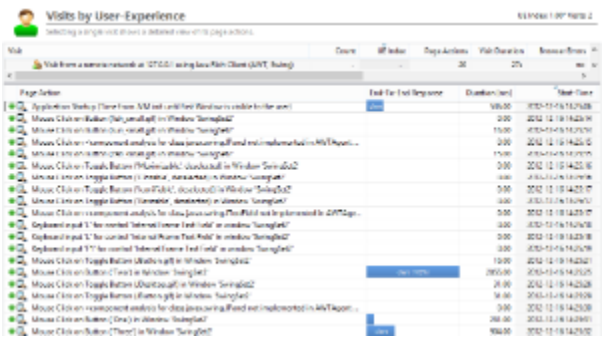
```
RichClientADK.CompuwareUEM.Shutdown();
```

Sample for Java AWT

Samples and Step-by-Step Guide for Java AWT

Start by downloading the AWT Sample Application: AWT_RichClient-UEM-ADK-Demo-src.zip
The sample application is already instrumented with the Rich Client UEM ADK which will create Page Actions for every AWT Control Interaction. Check out the following screenshots to get a better understanding on what information is captured by dynaTrace when interacting with the sample application:





Using the ADK on your own application

Step 1: Initialilze ADK
The only thing you need to do in your application is to call the initialize method of the AWT Agent. Here is an example:

```
/**
 * SwingSet2 Main. Called only if we're an application, not an applet.
 */
public static void main(String[] args) {
// ADK ////////////////////////////////////////////////////////////////
AWTAgent.initialize();
// ADK ////////////////////////////////////////////////////////////////
```

The initialize method will register an AWT Event Queue Handler. The handler will then create a page action for the most common window, mouse and keyboard events