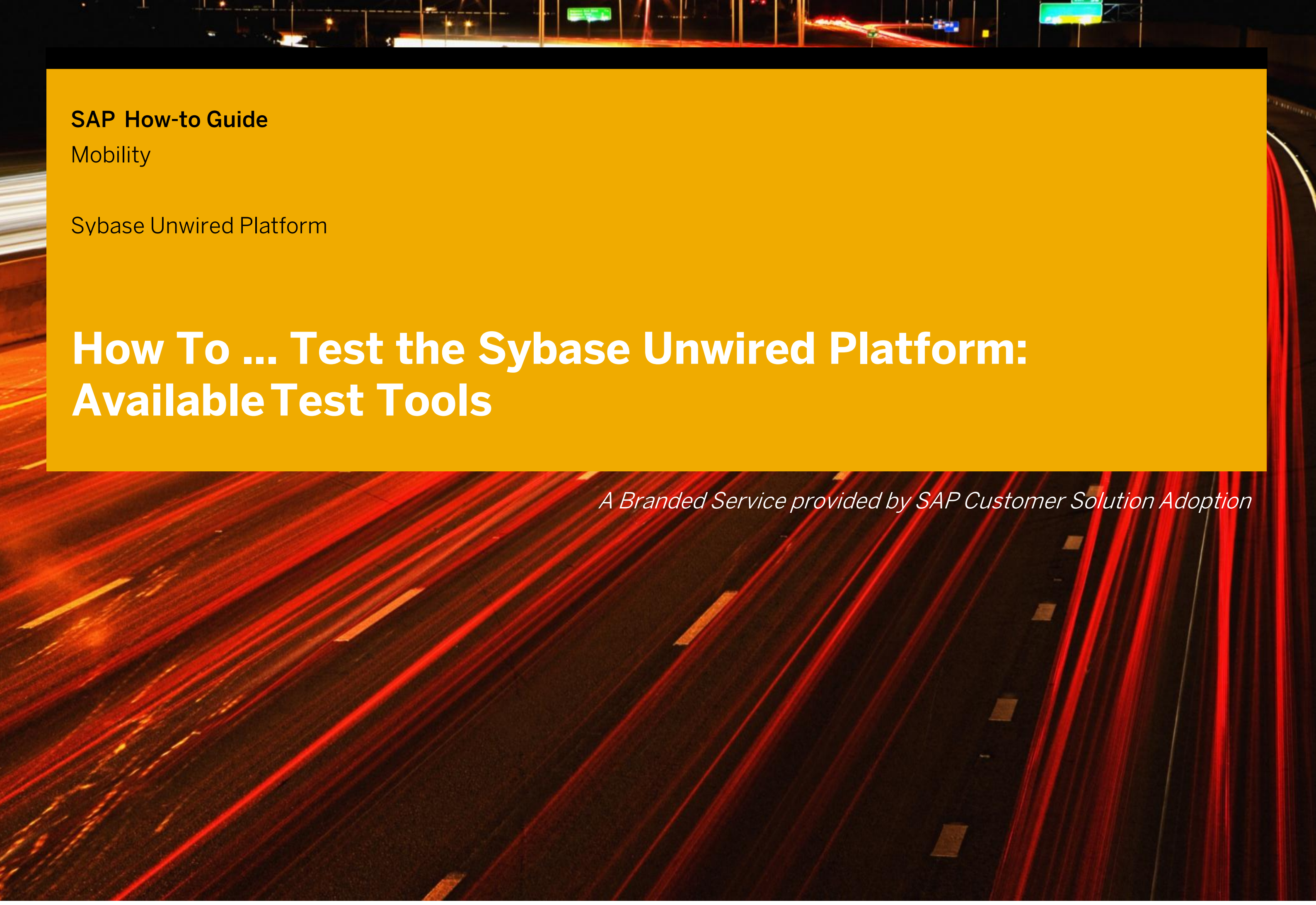


SAP How-to Guide

Mobility

Sybase Unwired Platform

How To ... Test the Sybase Unwired Platform: Available Test Tools

A Branded Service provided by SAP Customer Solution Adoption

Applicable Releases:

Sybase Unwired Platform 2.1

Version 1.0

March 2012

The Best-Run Businesses Run SAP™

Document History

Document Version	Authored By	Description
1.00	Patrick Kelleher	First release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Icons

Icon	Description
⚠	Caution
💡	Note or Important
❖	Example
⬆	Recommendation or Tip

Table of Contents

1. Business Scenario

This document is intended to explain the current tools available for testing mobile applications built using the Sybase Unwired Platform (SUP). The intent is not show how to test mobile apps but rather describe which tools can be used and what types of testing can be performed based on the type of mobile application.

The focus of the testing will be on the performance and reliability of the SUP, not the client application running on the mobile device.

2. Background Information

There are several different test methodologies, each have their own goals, procedures, and tools. In this document we will look at three areas of testing: functional testing, load testing, and performance measurement.

Functional testing

The goal of functional testing is to ensure the application or program works according to the user requirements. For our purposes we will include unit testing, integration testing, and end-user testing.

Functional testing usually starts with a test case. In unit testing this the test class the developer writes to test a unit of code. In higher-level functional testing, the testers have to follow a test case that describes the steps they must follow in the user interface of the application and what the outcome should be.

The Integrated Development Environment (IDE) used by the development team usually have frameworks for writing unit tests. Other functional testing is usually performed manually; however there exist test management software, such as SAP Solution Manager, that manages test cases and tracks errors.

Load Testing

The goal of load testing is to see if the application or system will fail under load. It is similar to performance testing in that you want to see how the system performs, and it is similar to functional testing in that you are looking to see if there are bugs or poor implementation aspects that cause the system to fail at certain loads.

In preparation for load testing it is important to collect appropriate test data, in terms of both content and volume. Where possible, it is best to use production data to source the test scripts. Analysis then has to be performed on the number of users who are expected to use the system or application, their geographical location, and the expected volumes the system will have to handle during its peak usage intervals.

The result of the preparation should be a set of test scripts that will simulate multiple concurrent users running different test cases and varying volumes of data. Each script might be written manually by a tester; or it might be generated by recording the data and input entered when a tester runs though a functional test case. LoadRunner is an example of a test tool that allows you to record user input into a web form or a SAP GUI interface in this manner.

Load testing also requires a test framework that allows you to run multiple scripts at once on at intervals, thus controlling the number and type of simulated concurrent tasks performed on the system. There are many commercial and public test frameworks available that perform this function.

Performance Measurement

The goal of performance measurement is to measure the system throughput while under load. Thus, performance measurement has the same requirements as load testing only the focus is on measurement and not functionality. The performance criteria that are measured during performance measurement are called metrics.

The preparation of the test scripts should follow from the load tests. In addition, agents will need to be installed and configured to measure the performance metrics. These agents might be part of the operation system, such as PerfMon; or they might be from a 3rd party tool, such as Wily Introscope.

Characteristic	Functional Testing	Load Testing	Performance Measurement
Goal	Ensure the Program/Application/ System works as intended	Ensure that the system works under stress	Measure the performance of the system under stress and determine performance bottlenecks
Procedure	Write and execute unit tests and test cases	Prepare test scripts and execute tests using load testing framework	Configure monitoring agents prior to running load tests
Tools	• Testing and debugging tools or the IDE. • SAP Solution Manager to manage test cases	• Recording tools for generating test scripts • Parsing tool to parameterize test script • Test framework that simulates load and collects results	• Agents to measure performance metrics • Framework to manage results • Analysis tools to interpret results and identify bottlenecks and points of failure.

Table 2-1: Overview of the types of testing methods and associated tools

3. Prerequisites

As the intent of this document is to provide information on the tools available for testing Mobile Application developed with SUP, there are no hardware or software requirements as such.

However, the reader should be familiar with the architecture of the Sybase Unwired Platform (http://www.sybase.com/detail?id=1095847), and should understand how performance is measured in SUP (http://www.sybase.com/detail?id=1096051).

4. Available Test Tools for testing SUP Mobile Apps

Before we look at how to test mobile applications built with Sybase Unwire Platform, we need to understand architectural types of SUP-based mobile apps. The SUP documentation differentiates between two basic arch-types: mostly online apps and mostly offline apps. Mostly offline or disconnected apps are defined by their cache. This cache is in sync with the SUP cache (CDB) which needs to stay in sync with the EIS source system.

For our purposes we will define three types of mostly offline apps: apps that use MBS (Message Based Synchronization), apps that use the DOE (Data Orchestration Engine) and apps that use RBS (Replication Based Synchronization). Technically DOE based apps use MBS but their server side cache resides in the DOE and not in the CDB of SUP.

For mostly online apps we'll assume Sybase Workflow and Hybird Web Container (HWC) apps are essentially the same. We will also include OData based apps in this group.

	Mostly Offline (Cache Based) Mobile Applications			Mostly Online Mobile Applications	
Technology	RBS	MBS	DOE	HWC	OData
Synchronization	Yes			No	
Mobile Data	Sybase UltraLite	SQL Lite		SQL Lite (XML)	Native App Persistence
Inner Protocol	MobiLink	iMO-Traveller			
Compression	ZLib				
Encryption	RSA	AES 128 – AES 1024			
Wire Protocol	Plain HTTP	Encapsulated Binary over chunked HTTP			
SSL	Yes	No			

Table 4-1: Mobile Application types and their associated technologies[1]

[1] Copied from the „SUP Development Best Practice" whitepaper by Martin Augst

4.1 Functional Testing Tools

Functional testing usually takes place on the device. The only exception is unit testing and debugging which is usually done in the IDE.

Device	IDE	Simulator
Win32, Windows Mobile	Microsoft Visual Studio	
iOS (iPhone, iPad)	XCode (Mac only)	Built-in
Android	Android SDK (eclipse based) http://developer.android.com/sdk/	Built-in
Blackberry	(eclipse based)	

Table 4-1-1: IDE's for popular mobile platforms

Additional tools for monitoring data transfer, debugging HTML/JavaScript, testing OData channels, and viewing cache data are also available.

Purpose	Tool	
SQL Lite	Firefox Add-on	
SQL Anywhere/Advantage DB	Sybase Mobile SDK	
Weinre	HTML5/JavaScript debugger	http://phonegap.github.com/weinre/
Firebug	Firefox Add-on (HTML5 debugger)	http://getfirebug.com/
Wireshark	Protocol analyzer	http://www.wireshark.org/
RESTClient	RESTful web service tester	https://github.com/chao/RESTClient

Table 4-1-2: Additional test tools for mobile application development

The Sybase Mobile SDK (eclipse based, formerly called the Unwired Workspace) offers tools to test connections to various back-end EIS systems and view the results of test calls. It also provides drivers for the Advantage and SQL anywhere database systems so you can view the contents of the Message Server queue and the CDB.

The ABAP workbench provides an IDE for testing and debugging OData Channel development in NetWeaver Gateway, and GET operations can be tested with any browser with a connection to the Gateway server. Other REST operations can be tested with a REST client such as the RESTClient Firefox Add-on.

HTML5 based OData clients can be debugged using Firebug or Weinre. These can also be included in HWC based libraries for debugging (follow this link to see how to include Weinre can be included in a HWC app: http://blogs.sybase.com/mobiledevelopment/2011/09/video-debugging-in-hwc-using-weinre/)

4.2 Load Testing Tools

In order to load test mobile applications you would need to control multiple instances of the application running at once. There are several options:

- Run the app manually on multiple mobile devices simultaneously (Gymnasium method)
- Run the app on multiple virtual devices simultaneously
- Emulate the app and run multiple instances on multiple virtual or emulated devices
- Emulate the data (of multiple app instances running on multiple devices simultaneously) being sent to, and record the data being received from, the mobile platform (SUP)

The first two options are only relevant if the expected peak usage of the app is low (< 200). Another consideration is: is really necessary to perform load tests on the device? Device characteristics might affect the app performance on the device and thus influence the frequency and duration of the connections and the amount of data carried across each connection (we'll refer to this as traffic).

Nevertheless, this variance can be profiled in functional testing. The device itself is mostly a single user device; therefore, load testing the device is not required.

If we omit the device from the testing then it should be enough to emulate the load on the SUP. The last two options propose two ways of emulating the test data:

1. Write test scripts that emulate the app (app centric).
2. Write test scripts that emulate the data traffic (data centric)

The app centric approach is simpler but more effort. This requires writing a functional test program that works according to the business rules which govern the mobile application. The test program may be the app code or a self-contained section of the app code however it must conform to the test harness.

The data centric approach requires a more complex approach. The wire protocol used by the mobile app must be understood. A tool is required that is able to record or capture the data of a sample test (execution of a particular use case of the app). The recording is then used to replay the wire protocol with other parameterized data, both consecutively and concurrently as determined by the test harness.

Both approaches require a test harness or framework. The test harness must be able to start and manage multiple instances of the test program/script, collect log entries, manage connections to the SUP server, and signal errors when they arise. It must also be able to control the data set used for testing and be able to control how data is fed to the various test instances.

The following table summarizes the steps required for each approach:

	App Centric	Data Centric
Steps	1. Set-up and configure test harness	1. Set-up and configure test harness
	2. Collect realistic test data (from a productive environment if feasible) based on the number of	2. Collect realistic test data (from a productive environment if feasible) based on the number of

	concurrent tests you plan to run.	concurrent tests you plan to run.
	3. Generate code stubs	
	4. Determine the use case for each test	3. Determine the use case for each test
	5. Set log levels to log as much as possible	4. Set log levels to log as much as possible
		5. Start recording/capturing all the data sent to the server
	6. Run the use case for each test on a test device with the actual app	6. Run the use case for each test on a test device with the actual app
	7. Use the use case, the business process description and the information collected from the logs to code the test scripts	7. Create scripts that parameterize fields in the recorded data. (The test harness will substitute test data for the parameterized fields.)
	8. Replicate the device conditions on the test platform	
	9. Functional test the test scripts	8. Functional test the test scripts
	10. Create a baseline for each test	9. Create a baseline for each test
	11. Ramp-up the tests	10. Ramp-up the tests
Issues	Implementation specific	Testing restricted to data not app
	Requires custom coding	Need to disable encryption and compression
	Difficult to maintain if the app changes	Test could break with new SUP version as it uses unpublished internal protocols
	Steep learning curve	Requires Sybase development to enable a scripting interface
	Test traffic may differ for actual application	Currently not practical for cache-based mobile applications
	Requires Sybase to expose method for setting the device id of the test client instance	Requires test framework to retrieve, store and replay surrogate keys
	Both approaches require a concept to handle push notifications (DCN)	
	Both approaches require a test user management utility to automate SUP device connection administration	

Load (or stress) testing tools currently available for testing SUP vary in their applicability to what they can test and where they can be used. Sybase has internal tools for load testing and a test harness developed for Sybase Professional Services to use for Sybase customers.

	Description	Tested	Technology	Commercial Use
HP LoadRunner	High-performance test framework for load testing enterprise applications	At least one customer project has used this to test SUP	RBS MBS OData HWC	SAP has special licensing for LR but this does not cover the .Net test license
JMeter	Open source test framework for testing enterprise applications	No information to date	App Centric approach only	Commercially available
ZAP-fix	Commercial mobile testing tool	No Information	App Centric	Commercially available
Perfecto MobileCloud	Commercial mobile testing tool	No Information	App Centric	Commercially available
Floodgate	Test Framework developed by Sybase for Sybase Professional Services (Consulting)	Used in several SUP projects	App Centric but could be adapted to Data Centric	Currently only available via SPS
Benchmark Application Framework (BAF)	Test Framework developed by Sybase for internal performance testing and measurement	Used internally	App Centric used primarily for RBS and MBS. Not tested with OData or HWC	Not commercially available
MobiLink replay utility (mlreplay)	Command line program used to "replay" RBS data transfers.	Used Internally	RBS only	Not bundled with SUP but available as a download from Sybase

HP LoadRunner

Load Runner is licensed through SAP for testing ABAP based systems and Web (HTML) interfaces. Currently there are only two methods for testing SUP with LoadRunner:

1. Custom C# program that consumes the MBO objects as described in the App Centric approach described above. This requires a .Net license for LoadRunner that is not part of the licensing agreement sold by SAP. SAP is working with HP to develop a licensing agreement for testing SUP.
2. Citrix client: this does not work for every app type but for MS Windows based apps Citrix can be used to record and parameterize the user interface of the app and replay the virtualized app for each test instance. This method works but requires a lot of hardware to simulate more than 100 client devices. The use of LoadRunner with the Citrix client is part of the SAP license.

4.3 Performance Measurement Tools

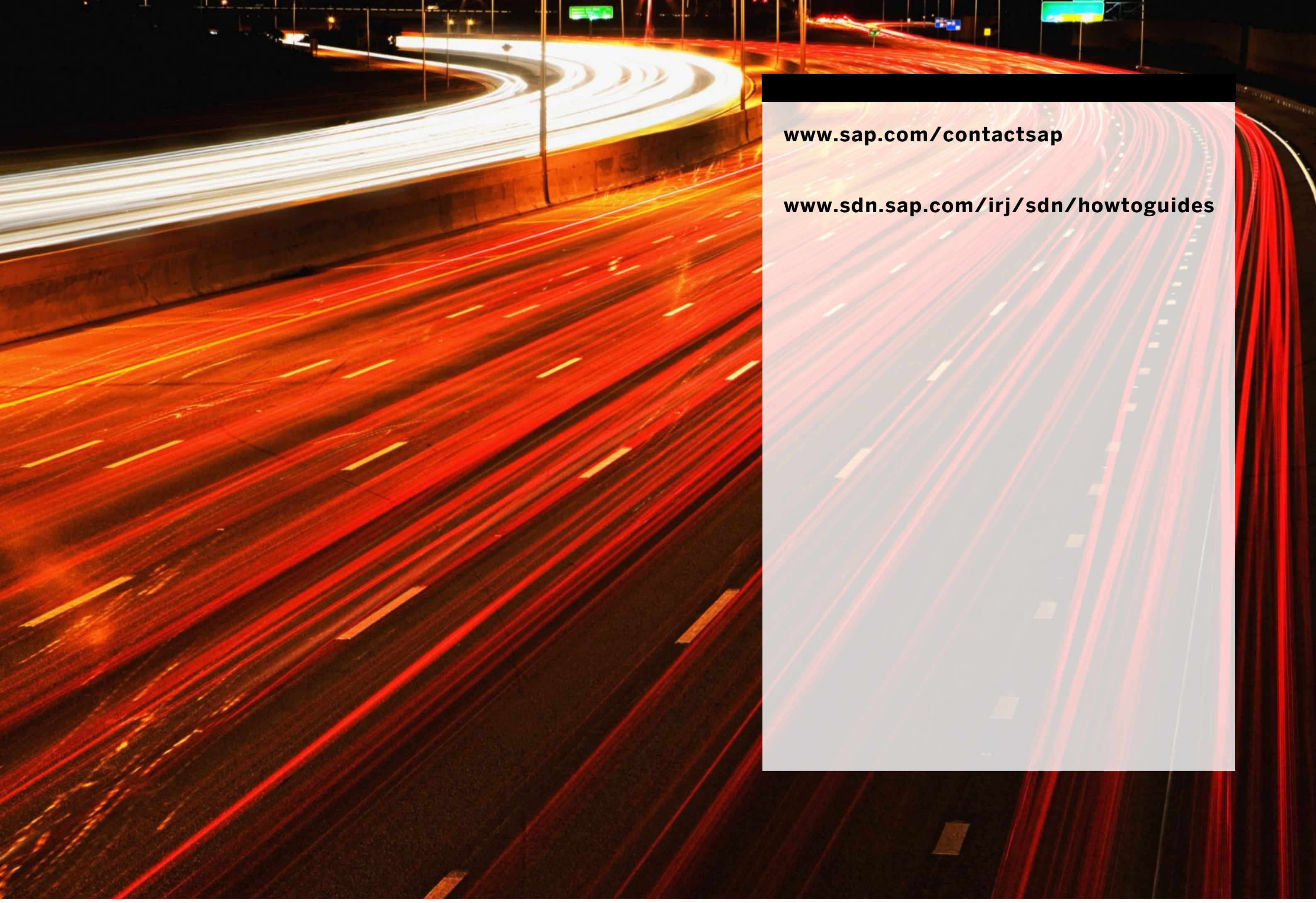
Performance measurement tools might be part of the operating system, part of the platform or might use hooks and or plug-ins into the software/platform in order to allow the tools to measure performance metrics of the software being tested. The following is a list of test tools that measure performance and performance measuring tools:

Purpose	Tool
Wily Introscope	Pure performance measurement tool that uses code injection (Aspect oriented programing) in Java to hook into the SUP to measure response times and other metrics.
HP Load Runner	Measurement instrumentation built into the test tool to monitor response times
BAF	Internal reports provide monitoring information on: • OS Level • SUP Server internals • User-defined statisitics
MLMon	Part of MobiLink, monitoring tool for the database replication that takes place during RBS synchronization.
PerfMon	Microsoft Windows operating system level monitoring for performance usage metrics of the system such as: • Network usage • CPU usage • Memory usage • File I/O

5. Appendix

References:

Tool	Reference
Microsoft Visual Studio	http://msdn.microsoft.com/en-us/vstudio
Windows Mobile Emulator	http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=9263
XCode	https://developer.apple.com/xcode/index.php
Android SDK	http://developer.android.com/sdk/
Blackberry	https://bdsc.webapps.blackberry.com/native/
SQLite	http://code.google.com/p/sqlite-manager/
SQL Anywhere/ Advantage DB	http://infocenter.sybase.com/help/index.jsp
Weinre	http://phonegap.github.com/weinre/
Firebug	http://getfirebug.com/
Wireshark	http://www.wireshark.org/
RESTClient	https://github.com/chao/RESTClient
HP LoadRunner	http://www8.hp.com/us/en/software/software-product.html?compURI=tcm:245-935779
JMeter	http://jmeter.apache.org/download_jmeter.cgi
ZAP-fix	http://www.zap-t.com/
Perfecto MobileCloud	http://www.perfectomobile.com/portal/cms/services/web_access_to_real_handsets.html
MobiLink replay utility	http://infocenter.sybase.com/help/topic/com.sybase.help.sqlanywhere.12.0.1/mlserver/ml-utilities.html
Wily Introscope	https://websmp207.sap-ag.de/~form/handler?_APP=00200682500000001943&_EVENT=DISPHIER&HEADER=Y&FUNCTIONBAR=N&EVENT=TREE&NE=NAVIGATE&ENR=01200314690900003513&V=INST&TA=ACTUAL&PAGE=SEARCH
MobiLink Synchronization monitor utility (mlmon)	http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc01205.0155/doc/html/aba1270578277892.html



www.sap.com/contactsap

www.sdn.sap.com/irj/sdn/howtoguides

The Best-Run Businesses Run SAP™