

ComponentArt White Paper:

# Delivering Mobile BI Dashboards to Any Device via XAML & HTML5



*Prepared by*

**The ComponentArt R&D Team**

Last Update: February 3, 2012

Version 1.2

# Table of Contents

- EXECUTIVE SUMMARY ..... 3**
  - TARGET AUDIENCE .....3
  - HIGH-LEVEL OBJECTIVES .....3
  - MOBILE DASHBOARDS IN ACTION.....3
- TECHNOLOGY OVERVIEW ..... 4**
  - WHY XAML AND HTML5? .....4
  - HIGH-LEVEL ARCHITECTURE.....5
  - CONNECTING TO DATA .....6
  - IMPLEMENTING DASHBOARD SCREENS .....6
  - SECURING THE APPLICATION .....6
  - DEPLOYMENT OPTIONS .....7
  - COMPONENTART PRODUCTS AND THEIR DEPENDENCIES.....7
- APPENDIX A – SUPPORTED DATA SOURCES AND MOBILE DEVICES ..... 8**
- APPENDIX B – INCLUDED DATA VISUALIZATION AND MOBILE UI CONTROLS ..... 9**
- APPENDIX C – GLOSSARY AND REFERENCE ..... 10**

## Executive Summary

### Target Audience

The purpose of this white paper is to provide a high-level technology recommendation for delivering versatile Business Intelligence dashboards to mobile devices. It is intended for both technical and business users, including BI Analysts, Project Managers, Systems Architects, and Software Developers. For further information please refer to in-depth technical documentation and detailed code examples which are included with ComponentArt's products.

### High-level Objectives

We are basing our technology recommendation on the assumption that successful mobile BI solutions need to be able to fulfill the following requirements:

1. **Leverage the existing BI infrastructure.** Organizations that have BI technology infrastructure in place (data warehouses, relational databases, administration tools, collaboration portals, security & authentication systems, reports, KPI definitions, etc.) will want to leverage their existing investment and build a mobile solution on top of it.
2. **Connect to any data source.** Having a standard and clean API for connecting to any existing or future data source is essential for the long term success of your mobile BI solution. This includes the ability to connect to standard "off the shelf" data sources, as well as any custom data providers that your technical team may develop.
3. **Reach any mobile device.** Even those organizations that currently standardize on a particular mobile platform would be wise to build a device-agnostic mobile BI solution, capable of reaching any modern mobile device through an industry-standard interface. The mobile device market is notoriously fast-changing. Today's leading vendors could quickly grow out of favor and be replaced by others, as evidenced by the dramatic market share fluctuations over the past couple of years.
4. **Deliver mobile dashboards securely.** Enterprise data needs to be delivered through secure mechanisms at all times. Utilizing a unified security and authentication infrastructure is paramount when delivering mobile applications. In addition, any fully secure mobile BI application must account for device loss.
5. **Deliver rich, interactive, touch-centric and screen form factor-friendly UI.** Truly effective and engaging mobile BI dashboards are designed for mobile devices from their inception and built using mobile presentation technology. Successful implementations will convey the key metrics effectively and "feel right" regardless of the screen size, orientation or the mobile platform in question.
6. **Build once, deploy anywhere.** In order to maximize development ROI and ensure ease of maintainability of your mobile solution, your development technology must be able to target multiple mobile platforms with a single codebase.

### Mobile Dashboards in Action

Before we dive into the technology recommendation, we invite you to experience live examples of finished dashboards that bring all of these elements together: <http://mobile.componentart.com>.

## Technology Overview

### Why XAML and HTML5?

When it comes to presentation layer technology, there are essentially three options for delivering BI dashboards to mobile devices. We will examine these options, comment on their pros and cons and explain why our recommended approach is XAML-to-HTML5 conversion:

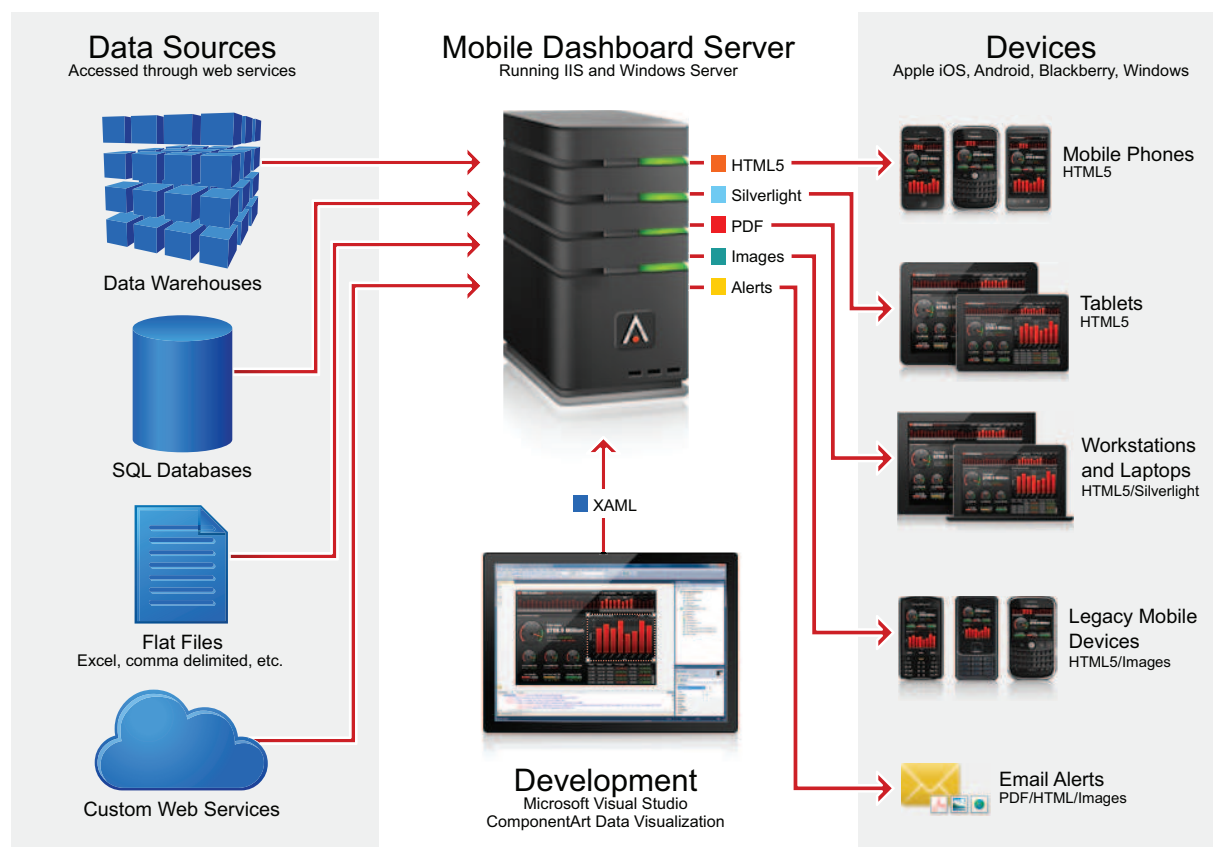
1. **Native Mobile Client Applications.** This option involves developing custom dashboard apps for each mobile platform individually. For example, your team would use the Apple iOS SDK (Software Development Kit) to build your iPhone/iPad mobile app. While this approach has potential to deliver great iPhone and iPad dashboards, all your implementation work would need to be repeated if you wish to target another mobile platform. All major mobile vendors have proprietary application development platforms, which are completely incompatible with each other:

Mobile Vendor	Development Library	Development Language
Apple	iOS SDK	Objective C
Google	Android SDK	Java
Microsoft	WP7+ SDK, Windows 8 WinRT	C#, VB.NET, JavaScript, C++
RIM	Blackberry SDK	Java

An alternative to building custom dashboard applications would be to license a third-party native mobile app that offers pre-packaged dashboard layouts which you could use to retrofit your data into. However, the complexity of developing for multiple mobile platforms is such that – to our knowledge – no such product supports all of the major mobile platforms listed above.

2. **Pure HTML5 Applications.** The advantage of HTML5 is that it is an industry standard – supported by all mobile vendors. Building an HTML5-based interface guarantees that it will be accessible not only by the current breed of mobile products, but also by future generations of mobile devices. HTML5 is quickly growing in popularity because it allows developers to build interfaces once and use them on any device. However, the feature set of HTML5 is in its infancy and is evolving very slowly due to the fact that it is driven by a consortium of competitors with unaligned interests. Subtle differences in feature implementations require developers to maintain tedious vendor-specific code branches. Finally, JavaScript as a programming language has limitations that prevent it from being the clear choice for complex, mission-critical applications.
3. **XAML-to-HTML5 Conversion.** XAML is the industry's most advanced presentation markup language. ComponentArt's XAML-to-HTML5 conversion technology aims to bring the best of both worlds: the vast XAML-based ComponentArt Data Visualization feature set (unreachable by pure HTML5 for several years, in our assessment), combined with the ubiquity of HTML5 on any device. The complexity of cross-platform HTML5 code generation is handled by our technology, while the developer creates the screens in modern XAML tools like Microsoft Visual Studio or Expression Blend. Finally, since XAML is a core building block of Windows 8, your mobile BI dashboards will be based on a technology of long-term strategic importance to the world's largest software company.

## High-level Architecture



The diagram above shows the high-level application architecture for delivering mobile BI dashboards based on ComponentArt's technology:

1. Dashboard screens are developed using modern XAML-based tools: Microsoft Visual Studio 2010 (or later) and the ComponentArt Data Visualization development framework. The XAML application can be based on Silverlight, WPF, or Windows 8 Metro / WinRT.
2. Data access is performed through a thin web service layer, typically developed in .NET or Java. Large organizations often have data warehouses that contain all of the data required by the dashboard/reporting system, accessed through OLAP queries. This scenario is obviously supported. However, any other combination of data sources is also supported, including SQL databases, flat files or custom web services.
3. Dashboard application(s) and data access web services are then deployed to ComponentArt Mobile Dashboard Server, running on Microsoft Windows Server and IIS. Deployment settings are customized, including: security and authentication, load balancing and output caching.
4. When a request is made, ComponentArt Mobile Dashboard Server instantiates the requested application screen, performs data access and converts the resulting screen output to HTML5 (or one of the other supported formats, if appropriate).
5. As the mobile user interacts with the screen, their actions result in state changes and new data binding operations. Each action generates new output that is sent to the mobile device.

## Connecting to Data

All data access is performed by implementing a simple web service with the sole purpose of connecting to the data source and fetching the data to the dashboard screen. The recommended approach is to implement the data access logic in .NET as either WCF or REST web services. These web services can be deployed to the same machine that is running ComponentArt Mobile Dashboard Server, or can reside on a separate machine. The only requirement is that the dashboard application running on the Mobile Dashboard Server machine is able to access these web services through a standard .NET API.

This approach essentially means that data access is accomplished through standard and proven .NET techniques that ensure availability of a vast number of possible data connectors (e.g. any data source that has an ODBC driver). Please see Appendix A for the list of supported data sources.

## Implementing Dashboard Screens

Dashboard screens are implemented in Microsoft Visual Studio 2010 or later, using standard XAML and .NET development techniques. The recommended application architecture is based on the Model View ViewModel (MVVM) design pattern. Following this design pattern ensures that application logic implemented in XAML and .NET will be successfully converted to HTML5 and JavaScript by the Mobile Dashboard Server runtime – without the need to write any JavaScript or HTML5/CSS code.

ComponentArt provides an extensive set of Data Visualization and Mobile UI controls (please see Appendix B for the full list), as well as a set of Mobile Dashboard Server project templates for Visual Studio. All included controls feature “HTML5 renderers”, designed to output interactive HTML5 content, optimized for touch-based interfaces and mobile screens. For example, the ComponentArt XyChart control has an HTML5 renderer capable of translating the XAML instance into HTML5 canvas-based output with touch interactivity for data points. This is handled automatically by the Mobile Dashboard Server runtime – the developer only needs to work with the XAML version of the control.

## Securing the Application

A ComponentArt Mobile Dashboard Server solution is deployed as a standard Internet Information Services application and secured using standard and proven IIS security techniques:

1. Authentication settings for the Mobile Dashboard Server application are set through the IIS Management Console or application configuration files. The supported methods are: Windows-based (Basic, digest, Integrated Windows Authentication), Forms or 3rd party authentication.
2. The Web.Config file of the Mobile Dashboard Server application is then configured to pass the user token information to the web service layer.
3. Mobile Dashboard Server security logic is implemented through provider-based architecture in the web service tier. The product ships with two security providers:
  - Active Directory Security Provider; and
  - Lookup Table Security Provider.

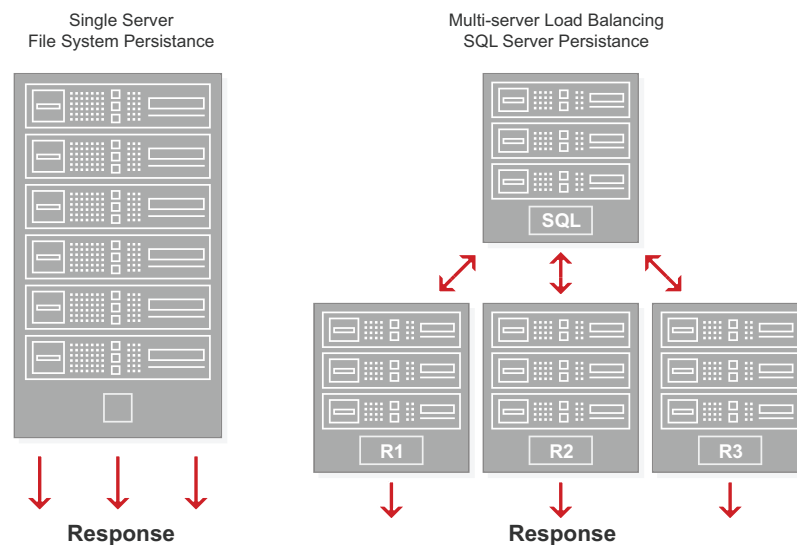
Customers can use the included security providers as a starting point for further customization.

## Deployment Options

ComponentArt Mobile Dashboard Server can be deployed in any hosting environment that supports Microsoft Windows Server and IIS. Therefore all of the following configurations are supported:

1. Physical servers, on-premise and behind the corporate firewall;
2. Virtual machines, on-premise and behind the corporate firewall;
3. Windows Azure cloud-based instances.

The product is architected to deliver Enterprise-grade performance through a comprehensive set of load balancing and output caching features:



ComponentArt Mobile Dashboard Server offers the following options for persisting output:

1. File system-based: for single-server deployments;
2. SQL database-based: for multi-server deployments with load balancing enabled. Under this configuration, multiple servers generate content and save it to a single SQL destination. The SQL destination could be a physical server running MS SQL Server, or Azure SQL storage.

ComponentArt Mobile Dashboard Server supports output caching for any renderable dashboard element, such as DashboardPanel or individual control (chart, gauge, grid, time navigator, etc.). Content expiration policies and invalidation rules can be set on any renderable dashboard element individually or the entire dashboard screen as a whole.

## ComponentArt Products and Their Dependencies

Product	Software Dependencies
ComponentArt Data Visualization	Microsoft Visual Studio 2010 (or later)
ComponentArt Mobile Dashboard Server	Microsoft Windows Server 2003 (or later) Microsoft .NET Framework 4.0 (or later)

## Appendix A – Supported Data Sources and Mobile Devices

List of Supported Data Sources:

Supported Data Sources	Supported Devices	Supported Output Formats
<u>File-based Data:</u> MS Excel CSV Files  <u>Relational Databases:</u> MS SQL Server Oracle MySQL MS Access  <u>OLAP Databases:</u> MS SQL Server Analysis Services IBM Cognos Oracle Essbase SAP BW  <u>Web Services:</u> SQL Azure MS SharePoint Lists Windows Azure DataMarket Custom WCF or REST Services	<u>Rich HTML5:</u> Apple iPhone/iPad Android phones and tablets Blackberry phones and PlayBook HP TouchPad Windows Phone 7.5+ Windows 8 tablets Windows & Mac OS computers  <u>Metro WinRT/Silverlight/WPF:</u> Windows 8 tablets Windows & Mac OS computers  <u>Flat HTML/images:</u> Legacy mobile devices (old phones and PDAs)	<ul style="list-style-type: none"> <li>▪ HTML5</li> <li>▪ HTML</li> <li>▪ Silverlight</li> <li>▪ WPF</li> <li>▪ PDF</li> <li>▪ PNG Image</li> <li>▪ JPEG Image</li> <li>▪ TIFF Image</li> <li>▪ BMP Image</li> </ul>

## Appendix B – Included Data Visualization and Mobile UI Controls

The following table contains the full list of controls included with ComponentArt Data Visualization and ComponentArt Mobile Dashboard Server Products:

Mobile Data Visualization Controls	Mobile UI Controls
<p><b><u>Charting</u></b></p> <ul style="list-style-type: none"> <li>▪ XyChart</li> <li>▪ PieChart</li> <li>▪ FunnelChart</li> <li>▪ RadarChart</li> <li>▪ TreeMap</li> </ul> <p><b><u>Gauges</u></b></p> <ul style="list-style-type: none"> <li>▪ RadialGauge</li> <li>▪ LinearGauge</li> <li>▪ NumericGauge</li> <li>▪ BulletGraph</li> <li>▪ Cylinder</li> <li>▪ HalfDonut</li> <li>▪ Thermometer</li> <li>▪ Indicator</li> </ul> <p><b><u>DataGrids</u></b></p> <ul style="list-style-type: none"> <li>▪ GridView</li> <li>▪ PivotGrid</li> </ul> <p><b><u>Advanced Data Visualization</u></b></p> <ul style="list-style-type: none"> <li>▪ Map</li> <li>▪ TimeNavigator</li> <li>▪ CalcEngine</li> </ul> <p><b><u>Common Controls</u></b></p> <ul style="list-style-type: none"> <li>▪ Legend</li> <li>▪ DashboardLayout</li> <li>▪ DashboardPanel</li> <li>▪ DrillDownManager</li> <li>▪ WcfDataProvider</li> </ul>	<ul style="list-style-type: none"> <li>▪ TextBox</li> <li>▪ CommandButton</li> <li>▪ LinkButton</li> <li>▪ RadioButton</li> <li>▪ CheckBox</li> <li>▪ Dropdown</li> <li>▪ Slider</li> <li>▪ RangeSlider</li> <li>▪ TreeView</li> <li>▪ DropDownTree</li> <li>▪ DatePicker</li> </ul>

## Appendix C – Glossary and Reference

Term	Definition and/or further information
.NET	<a href="http://www.microsoft.com/net">http://www.microsoft.com/net</a>
Active Directory	<a href="http://en.wikipedia.org/wiki/Active_Directory">http://en.wikipedia.org/wiki/Active_Directory</a>
API	<a href="http://en.wikipedia.org/wiki/Api">http://en.wikipedia.org/wiki/Api</a>
Authentication	<a href="http://en.wikipedia.org/wiki/Authentication">http://en.wikipedia.org/wiki/Authentication</a>
ComponentArt Dashboard Server	<a href="http://www.componentart.com/products/ds/">http://www.componentart.com/products/ds/</a>
ComponentArt Data Visualization	<a href="http://www.componentart.com/products/dv/">http://www.componentart.com/products/dv/</a>
ComponentArt Documentation	<a href="http://docs.componentart.com/">http://docs.componentart.com/</a>
CSS	<a href="http://en.wikipedia.org/wiki/Cascading_Style_Sheets">http://en.wikipedia.org/wiki/Cascading_Style_Sheets</a>
Data Warehouse	<a href="http://en.wikipedia.org/wiki/Data_warehouse">http://en.wikipedia.org/wiki/Data_warehouse</a>
HTML5	<a href="http://en.wikipedia.org/wiki/HTML5">http://en.wikipedia.org/wiki/HTML5</a>
IIS	<a href="http://www.iis.net/">http://www.iis.net/</a>
JavaScript	<a href="http://en.wikipedia.org/wiki/JavaScript">http://en.wikipedia.org/wiki/JavaScript</a>
KPI	<a href="http://en.wikipedia.org/wiki/Key_performance_indicator">http://en.wikipedia.org/wiki/Key_performance_indicator</a>
Load Balancing	<a href="http://en.wikipedia.org/wiki/Load_balancing_(computing)">http://en.wikipedia.org/wiki/Load_balancing_(computing)</a>
ODBC	<a href="http://en.wikipedia.org/wiki/ODBC">http://en.wikipedia.org/wiki/ODBC</a>
OLAP	<a href="http://en.wikipedia.org/wiki/Online_analytical_processing">http://en.wikipedia.org/wiki/Online_analytical_processing</a>
Relational Database	<a href="http://en.wikipedia.org/wiki/Relational_database">http://en.wikipedia.org/wiki/Relational_database</a>
REST Web Service	<a href="http://en.wikipedia.org/wiki/Representational_state_transfer">http://en.wikipedia.org/wiki/Representational_state_transfer</a>
SDK	<a href="http://en.wikipedia.org/wiki/Sdk">http://en.wikipedia.org/wiki/Sdk</a>
UI	<a href="http://en.wikipedia.org/wiki/User_interface">http://en.wikipedia.org/wiki/User_interface</a>
WCF	<a href="http://en.wikipedia.org/wiki/Windows_Communication_Foundation">http://en.wikipedia.org/wiki/Windows_Communication_Foundation</a>
XAML	<a href="http://en.wikipedia.org/wiki/Xaml">http://en.wikipedia.org/wiki/Xaml</a>