White Paper

embarcadero®

Mobilizing your Business with Enterprise Mobility Services Middleware

Preview

By Cary Jensen

April 2015

Table of Contents

Executive Summary 3
Introduction 3
Overview of Enterprise Mobility Services (EMS) 5
Continue Reading 7
Acknowledgements 8
About the Author 8

Executive Summary

The Enterprise Mobility Services feature of RAD Studio is a set of services that provide a turnkey solution for building multi-tier applications. Enterprise Mobility Services, or EMS, consists of two services. The first is a REST (REpresentation State Transfer) web service that supplies a ready-made administrative API (application programming interface), and which can easily be extended with custom endpoints. The administrative API provides REST endpoints that support the creation and management of users and groups. Custom REST endpoints permit you to selectively expose the features and data required by your client applications.

Because EMS uses REST, EMS features can be accessed from a wide range of client applications, from traditional desktop applications to mobile apps, from web pages using JavaScript to third party web services. Mobile clients, in particular, benefit from the ease with which data can be accessed from your EMS server, since these platforms typically lack the ability to connect to remote databases directly. And because all EMS endpoints can enforce authentication and authorization, you can control who can access your data and from where.

The second service provided by EMS supplies you with analytics that give you detailed information about which users and user groups are accessing your EMS service, and how. This service tracks user access by the hour, day, month, and year. It also reports which of your REST endpoints are called and when. This information can help you make decisions about which areas of your applications deserve your development resources, and which users are taking advantage of your services.

In addition to the two services already mentioned, your EMS license includes a license for Embarcadero's InterBase database server for secure enterprise SQL data storage that can be used by your EMS service, and InterBase ToGo licenses, which permit you to install secure, transaction-based databases on your mobile EMS clients.

Introduction

Enterprise Mobility Services (EMS) is a multi-tier framework that first shipped with RAD Studio XE7. This white paper is designed to provide you, the software professional, with the information you need to get started with EMS today.

This paper begins with a general overview of EMS. In this section you will learn how EMS interfaces with the various technologies necessary to support your applications, and why this approach is so valuable in today's distributed computing landscape.

The paper continues by showing you how to get started with your EMS development. This section begins by showing you how to install EMS and make basic calls into this service. Importantly, these calls are standard REST (REpresentational State Transfer) calls, which can be emitted by any language or framework that understands HTTP (HyperText Transfer Protocol). After a brief discussion of REST, you will learn how to make some of these basic calls using a browser, RAD Studio's REST Debugger, and a Delphi application.

Next, you will learn how to extend the provided administrative API (application programming interface) with your own custom REST endpoints, making the data and functionality of your service available to the broadest range of client applications. Here you will learn how to create EMS modules, define and implement EMS resources, and access these custom REST endpoints from EMS clients.

With the EMS server now sporting more than its administrative API, it's time to consider security. In this section you will learn how to require a user to identify themselves before they can access EMS resources, a process called authentication. In addition, you will learn how to enable authorization, a mechanism that can restrict particular resources and endpoints to select groups of users.

Up to this point, you will be working with the EMS development server, an HTTP server that ships with almost all versions of RAD Studio (The EMS developer edition does not ship with the Starter Editions of Delphi or C++Builder, and is available as an optional add-on to RAD Studio, Delphi and C++Builder Professional editions by purchasing the FireDAC Add-On Pack). The deployment versions of EMS servers are ISAPI DLL, which run in conjunction with IIS (Microsoft's Internet Information Services). In this section you will learn how to configure IIS 7 to enable the EMS ISAPI DLLs, and how to configure your client applications to access this version of the server. As a bonus, you will learn how to access this ISAPI DLL from JavaScript and jQuery, permitting you to build web sites that get their data from your EMS endpoints.

As software developers, we probably spend as much time debugging and maintaining our software as designing it. In this next section you will learn how to step through your custom endpoints in your EMS packages, both when using the EMS development server, as well as with the ISAPI DLL version.

In the final section of this white paper you will learn how EMS compares to another Embarcadero middle-ware solution – DataSnap.

Before we continue, I want to emphasize that this paper is about EMS technology, how to implement it, and how to access those implementations. It is not, despite the title, about developing mobile solutions but rather how to use EMS to enable remote data access and application logic. Even the name of this technology, Enterprise Mobility Services, could imply a mobile-only technology, which seems limited given the wide access EMS can provide to other clients.

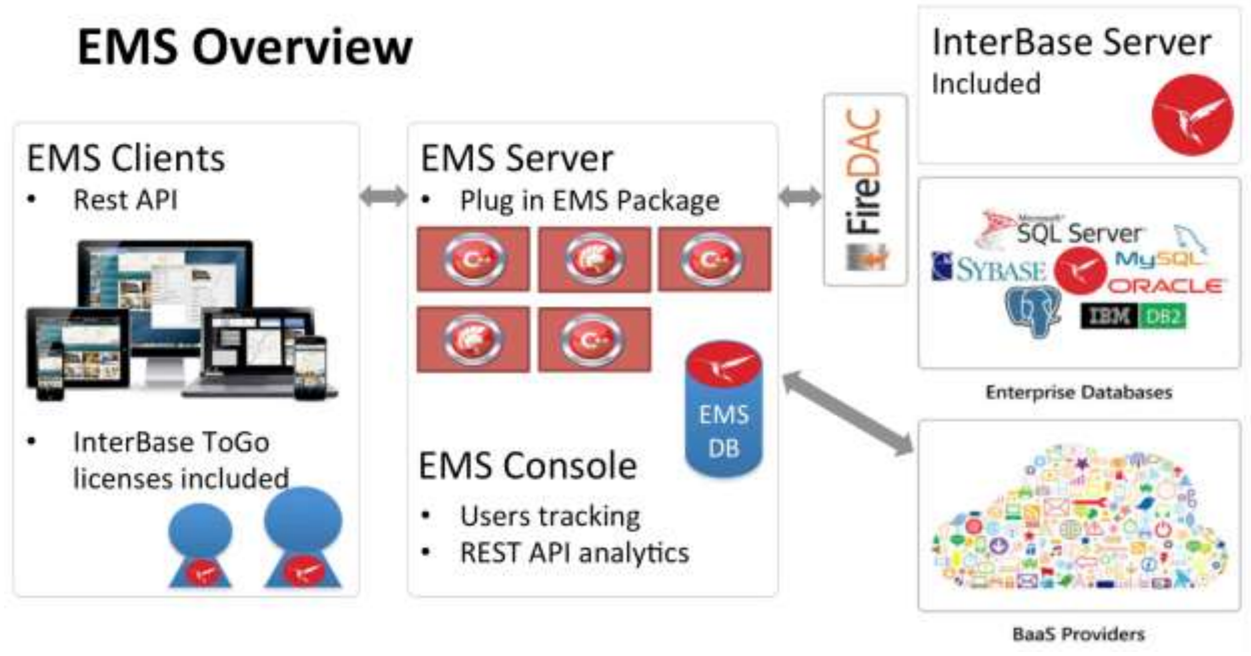
In truth, EMS is about highly accessible data, whatever the platform. It's about thin clients, those that do not, and in many cases, cannot, access enterprise data directly. In those cases, those thin clients must go through some sort of intermediary. EMS is such a solution, using a REST web service as the universally accessible platform for bi-directional data movement and remote procedure calls.

Because this paper is about accessing EMS data from client application in general, I have not focused specifically on mobile clients to demonstrate this access. Yes, the Delphi application that I use to demonstrate EMS data access can be compiled and deployed to an iOS or Android phone or tablet. However, this demonstration project was not designed as a mobile application, as these types of applications have user interface requirements and limitations that are beyond the scope of this paper. In other words, the techniques for accessing EMS endpoints that I demonstrate can and should be used in mobile applications, but I ask that you look to other white papers and video presentations on the Embarcadero site for details about mobile application user interface design.

Overview of Enterprise Mobility Services (EMS)

Enterprise Mobility Services is an extensible, turnkey, middle-ware platform for securely exposing data and features to a wide range of client applications. These data and features are exposed through a REST interface that supports authentication and authorization and encrypted client/server communication. EMS client applications can be traditional desktop applications, mobile applications, web sites and web services, and any other type of application that can consume REST services.

The following figure, which was taken from RAD Studio's online help, depicts a general overview of the EMS technology stack. At the heart are the EMS server and the EMS console, depicted in the middle rectangle of this diagram.



The EMS server is a REST web service that exposes a collection of pre-defined REST endpoints that permit the management of users and groups. These users and groups are stored in InterBase database, and an InterBase license for this database is included with the EMS license.

The EMS console is a web service that exposes essential statistics about the EMS database and EMS server usage. For example, the EMS console lets console users view the server's users and groups. The EMS statistics exposed by the EMS console include the number of unique user visits, number of REST API calls and their distribution among the available REST endpoint. These data can be summarized by hour, day, month, or year, giving administrators insight into how the EMS server applications are being used.

The EMS server can be extended by adding one or more EMS packages, modules that can be created using RAD Studio's Delphi or C++Builder languages. It is through these custom endpoints that you expose your organization's data and operations. For example, you can create a set of custom endpoints that permit your mobile clients to request data from your company's database. Another set of endpoints can permit these clients to send updates back to the EMS server, where your custom code can evaluate these changes, and selectively post the changes back to the central database.

Since these custom endpoints are written in C++ or Object Pascal, they can use all of the power of RAD Studio to perform their necessary tasks. This is represented by the arrows between the EMS Server and the entities in the right-hand rectangle in the preceding diagram. For example, you can use FireDAC, or any of the other data access mechanisms in RAD Studio to communicate with your enterprise database servers. And if you don't already have a SQL database server to store the data for your EMS applications, you can use the InterBase license that is included with EMS.

But it's not just databases that we are talking about. RAD Studio supports all of the standard protocols for network communication, and also includes a wide range of components that simplify your use of industry standard services. Your custom endpoints can provide your EMS clients with a gateway to all of your data, wherever it resides, whether it be in the cloud, inside your firewall, or even in local files on your servers.

Regardless of where your data is stored, your EMS server provides a secure gateway to you data. Your EMS server supports authentication and authorization, out of the box, along with SSL (Secure Socket Layer) encryption of your client/server communications.

But the real value of your EMS server comes from its clients. Because EMS exposes its endpoints using an industry standard REST API, any platform can host EMS clients. Using RAD Studio, you can build native mobile applications for iOS and Android, and these can use EMS client components to effortlessly interact with your EMS server. In addition, if you need secure local storage on your mobile devices, no problem. EMS includes licenses for InterBase ToGo, a small-footprint, transaction-supporting database server that can be installed on your mobile devices, and which supports encrypted storage.

You are not limited, however, to mobile applications. Your EMS client applications can be interactive desktop applications, Internet services, web services, console applications, in short, any type of application that can issue and consume HTTP requests.

Neither are you limited to EMS clients written in RAD Studio. For example, web pages employing JavaScript can access your EMS endpoints, so long as they abide by the security requirements of your EMS server, such as providing a valid username and password before being granted access to other EMS endpoints. In addition, logged in users will only be able to access the endpoints to which they are granted access rights. But that is a configuration issue, and you will learn how to do that later in this paper.

Taken together, EMS provides you with a secure platform from which you can access any data from anywhere.

Continue Reading

To read more, download the full paper at
http://forms.embarcadero.com/EMSMiddlewareWP

Acknowledgements

I want to express my thanks for the generous help of the many people who were involved in the production of this white paper and the associated Webinar. In particular, I want to thank Jim Tierney, Principle Engineer at Embarcadero Technologies, for his technical support, several code samples, and incisive input on a draft of this paper. Similar kudos go to Marco Cantú, RAD Studio Product Manager, for his technical support and review of the paper. I also want to thank John Thomas (JT), for asking me to write this paper, his marketing insight, and for ensuring that I got the technical support that I needed. I also want to thank my wife and business partner, Loy Anderson, who also reviewed the final draft of this paper. Finally, I want to thank the many other people at Embarcadero Technologies who helped this project at various stages, including Tim Del Chiaro, Brian Alexakis, David Intersimone (David I), and Jim McKeeth.

About the Author

Cary Jensen is Chief Technology Officer of Jensen Data Systems, Inc., a Texas, USA-based company that is an Embarcadero Technology Partner and that provides training, development, and consulting services. Cary is an Embarcadero MVP and an award-winning, best-selling author of over two dozen books on software development, including his latest book, Delphi in Depth: ClientDataSets 2nd Edition. He has written hundreds of magazine articles, and is a popular speaker at conferences, workshops, and training seminars around the world, and is widely regarded for his practical solutions to complex problems. Cary wrote, and was the principal speaker for the original Delphi World Tour in 1995, and in 2001 he founded Delphi Developer Days, a popular training seminar that visits cities in North America and Europe each spring. Cary has a Ph.D. in Engineering Psychology from Rice University, specializing in human-computer interaction. You can learn more about Cary at www.JensenDataSystems, and follow him on twitter (@caryjensen).

© 2015 Embarcadero Technologies, Inc. Embarcadero, the Embarcadero Technologies logos, and all other Embarcadero Technologies product or service names are trademarks or registered trademarks of Embarcadero Technologies, Inc. All other trademarks are property of their respective owners 090415