RAD SERVER IN DEPTH

Marco Cantu, Delphi Product Manager
marco.cantu@embarcadero.com
Twitter: @marcocantu

embarcadero®

AGENDA

- What is RAD Server
  - Returning JSON for JavaScript and Ext JS clients
- Exposing FireDAC database data
  - Streaming
  - Batch Move
  - MetaData management
- Users and Permissions
- Analytics and other Advanced Features

PART I:

RAD Server Introduction

WHY RAD SERVER?

- Rapidly connect your apps to your enterprise databases and services hosted on-premises or in a private cloud
- Make enterprise data easily available on every device, keeping it secure
- Includes user and groups usage analytics, along with REST API calls analytics

Application Services

KEY BUILT-IN CORE SERVICES TO POWER YOUR APPLICATION

RAD Server includes a powerful set of built-in core servers to power your application back-end eliminating the need to build the key foundational components of your server application. Core services like User directory services, authentication, and access control, push notifications, JSON data-storage, and user proximity and indoor/outdoor user location tracking and fencing.

Push Notifications

Send programmatic or on-demand notifications to your application users.

Built-in Secure Datastore

Easily store and retrieve JSON data securely and without requiring a separate database server.

User/Group Management

Create and manage users, groups, and access control via the RAD Server management portal.

User Location/Proximity

Track user movement both indoors and outdoors, and respond to proximity events when users enter or exit custom beacon zones or approach designated beacon points.

REST End-Point Publishing

EASY REST API END-POINT CREATION, PUBLISHING, AND MANAGEMENT

RAD Server makes it fast and easy to build flexible back-end servers for your multi-tier application solution. Developers simply load Delphi and C++ business logic into the server, and managed REST/JSON API end-points are generated. Developers or admins can easily configure API level access control to user groups, and measure and analyze application usage at the API, user, or service level. Since your Delphi and C++ APIs are published as heterogeneous REST/JSON end-points, RAD Server easily supports virtually any client type from VCL and FMX Desktop, Mobile and Wearable clients to popular JavaScript Web Frameworks. RAD Server easy to build and deploy robust heterogeneous multi-tier solutions.

{API}

REST End-Point Publishing

Easy to use API publishing of business logic. Any Delphi or C++ code can be hosted as an API and auto-published as REST/JSON endpoints which are measured and managed by RAD Server.

API Analytics

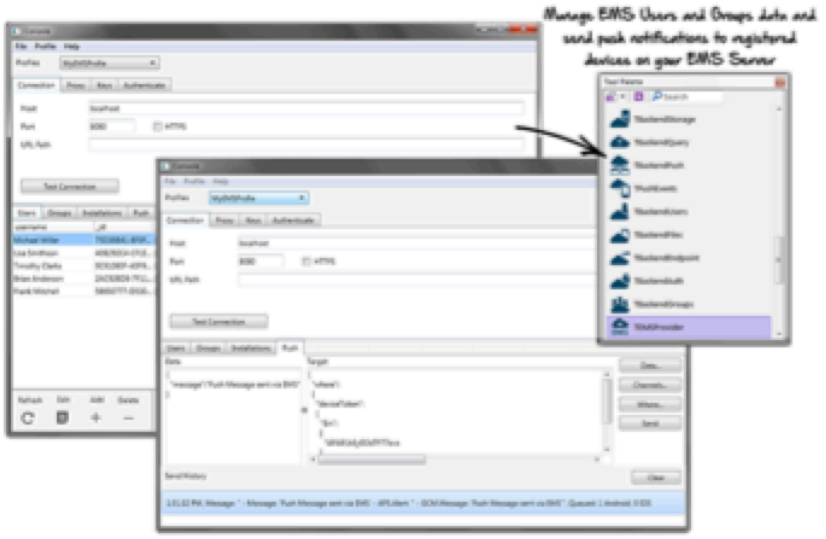
All REST API end-point activity is recorded and measured for robust statistics tracking and analytics. Analyze user, API, and services activity to gain insight into how your application is being utilized.

Access Control

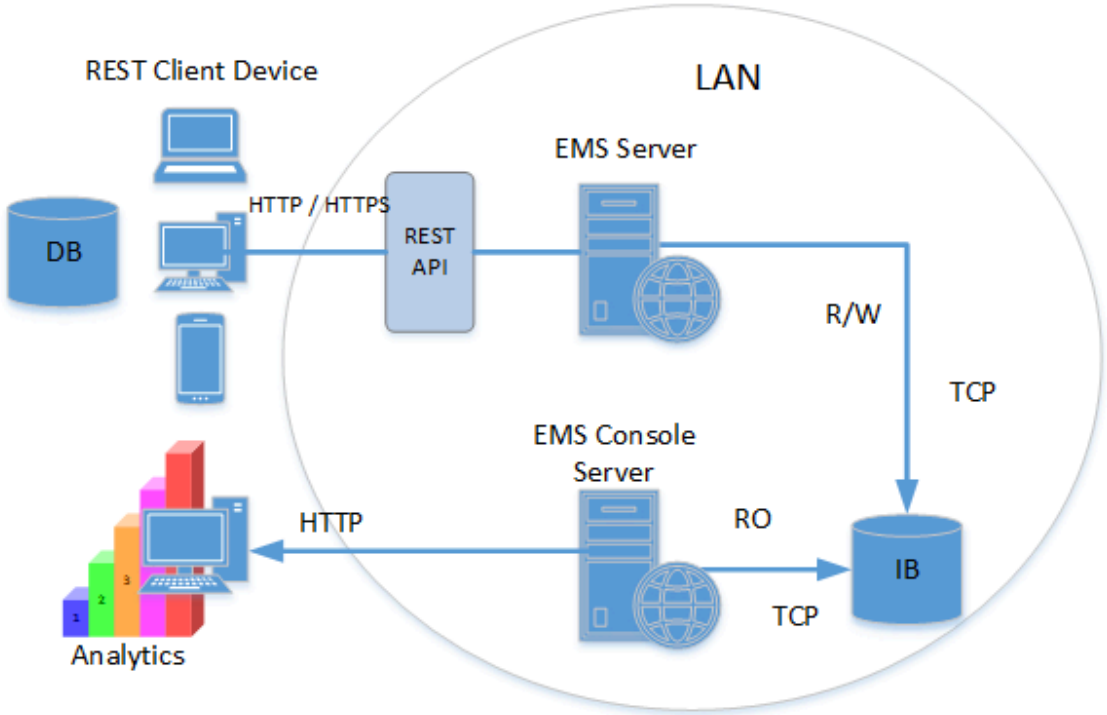
Group and user level access to control to all application APIs. Control who has access to what functionality. All access is user authenticated.

Desktop, Mobile & Web

All C++ and Delphi code hosted on RAD Server are published as REST/JSON end points consumable by any type of client for extreme flexibility and future-proofing.



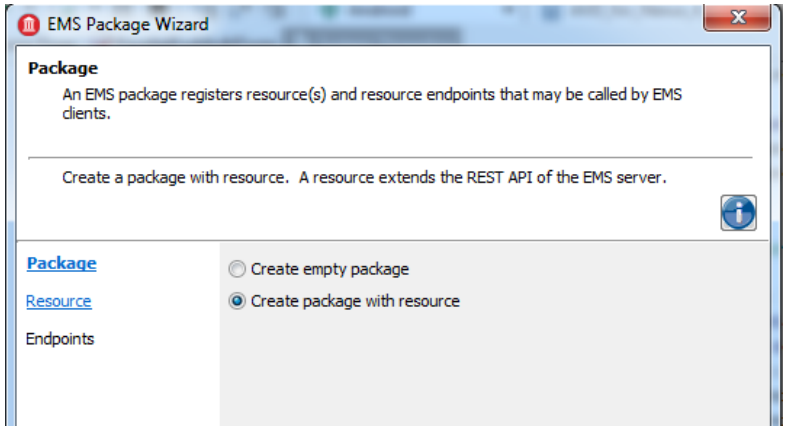
RAD SERVER ARCHITECTURE



FIRST STEPS IN PUBLISHING APIS

- Creating an EMS resource package
  - Implementing the REST APIs
  - Executing the server
  - Calling the APIs form a browser or a client app
- Examining the console
  - And the user management app
- Configuration wizards
  - And the EMS.INI file key settings

RAD SERVER PACKAGES



- Add resources to RAD Server
  - Mapped to URI
- Wizards help create packages and add a resource to an existing package
  - RAD Server Package Wizard, optionally adds first resource
  - RAD Server Module Wizard, for more resources

RAD SERVER CLIENTS

- Experimenting with a browser
- Using the REST debugger
- Clients make REST API calls
  - Any HTTP library would work
- BaaS architecture for the client side
  - EMS Provider component to connect
  - Use BAAS Components for services
    - TBackendQuery, TBackendUser, TBackendEndpoint
  - Handy TEMSFireDACClient component
    - For transferring data and delta packets between client and EMS server

RESOURCES AND ENDPOINTS

- Get and GetItem
- Put, Post, Delete
- Query params and other HTTP request information

- Also, return files from folders, host HTML + JavaScript apps

RAD SERVER AND FIREDAC

- FireDAC JSON Streaming
  - SaveToStream, LoadFromStream with *sfJSON* format
  - Expose as resource, consume via BackendEndPoint
- Using the BatchMove architecture
- Use FDSchemaAdapter
  - Captures master/detail relationships on server
  - Exposes the same on the client
  - Direct hooks to streaming, invocation

USERS AND PERMISSIONS

- Basic users management
  - User endpoints
  - Users authentication
  - BackendAuth component
- Permissions
  - Users authorization
  - Groups

ADVANCED FEATURES

- Built-in user and API analytics
- Advanced users management and custom login modules
- Integrated push notifications support
- EdgeModules or ThingPoints
- Multi-tenancy

DOCUMENTING YOUR APIS

- Automatic doc support
- Publishing custom API endpoints docs
  - Swagger RESTful API Documentation Specification
  - JSON or YAML formats

THANKS!
Any questions?

You can find me at:
@marcocantu
marco.cantu@embarcadero.com