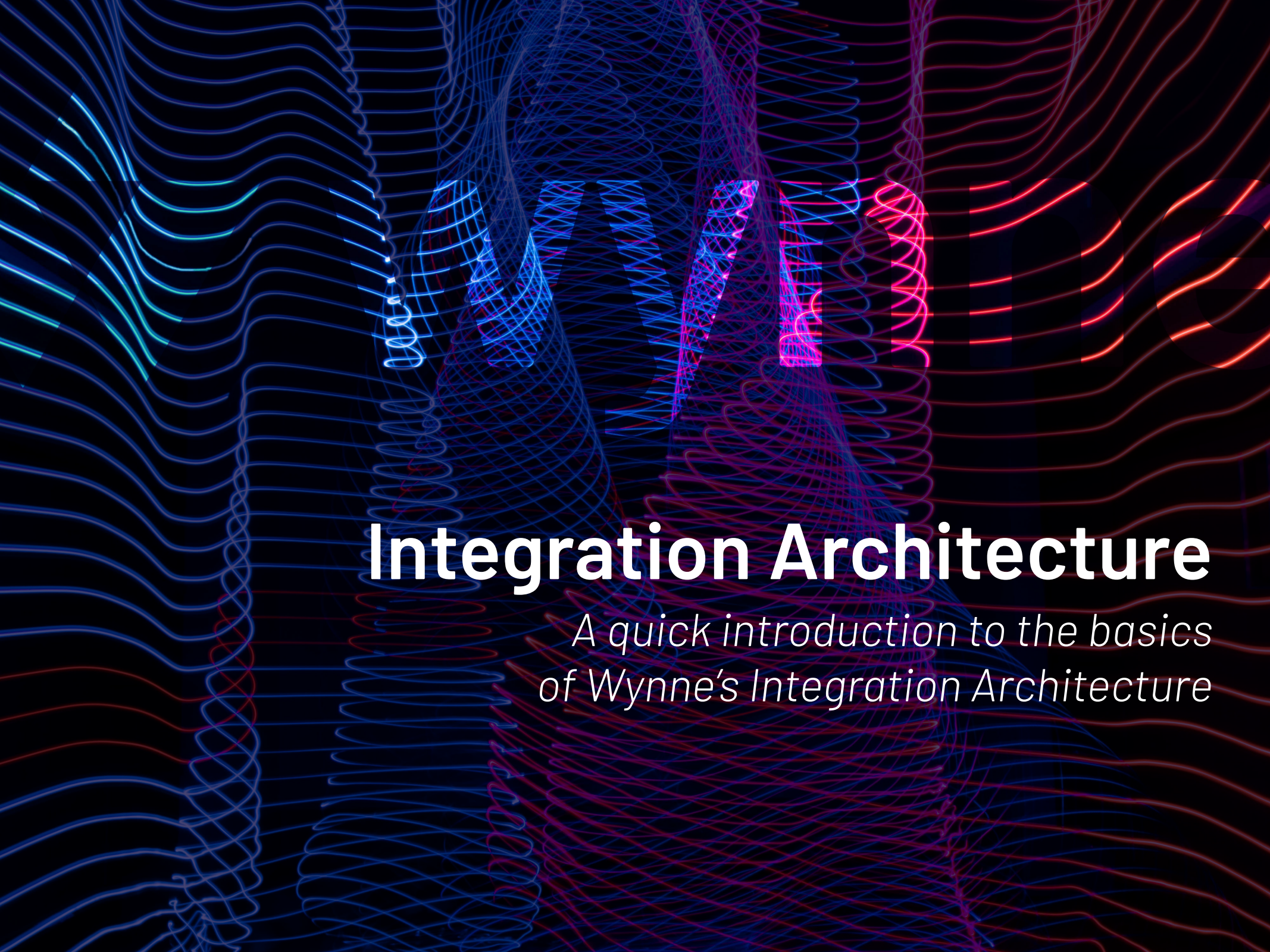


Integration Architecture

A quick introduction to the basics of Wynne's Integration Architecture

The Basics: What is XML?

→ The biggest difference between is that **HTML describes presentation** and **XML describes content**

→ An HTML document rendered in a web browser is human readable

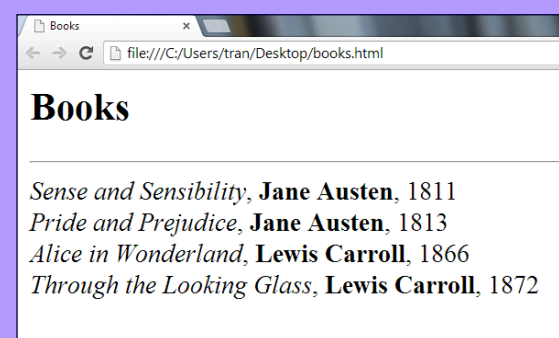
→ XML is aimed towards being both human and machine readable

The Basics: XML vs HTML

HTML Example

```
<html>
<head><title>Books</title><head>

<body>

<h2>Books</h2>
<hr>

<i>Sense and Sensibility</i>, <b>Jane Austen</b>, 1811<br/>
<i>Pride and Prejudice</i>, <b>Jane Austen</b>, 1813<br/>
<i>Alice in Wonderland</i>, <b>Lewis Carroll</b>, 1866<br/>
<i>Through the Looking Glass</i>, <b>Lewis Carroll</b>, 1872<br/>

</body>
</html>
```

Rendered Code



The HTML describes how the information is to be presented and formatted for a human to view in the web browser. Knowing that **Sense and Sensibility** is enclosed in italic tags does not help a program determine that it is the title of the book.

The Basics: XML vs HTML

XML Example

```
<books>
  <book>
    <title>Sense and Sensibility</title>
    <author>Jane Austen</author>
    <year>1811</year>
  </book>

  <book>
    <title>Pride and Prejudice</title>
    <author>Jane Austen</author>
    <year>1813</year>
  </book>

  <book>
    <title>Alice in Wonderland</title>
    <author>Lewis Carroll</author>
    <year>1866</year>
  </book>

  <book>
    <title>Through the Looking Glass</title>
    <author>Lewis Carroll</author>
    <year>1872</year>
  </book>
</books>
```

A program parsing this data can take advantage of the fact that all book titles are enclosed in <title> tags.

The Basics: Web Services

- Definition (from Wikipedia):
- "Web services are typically application programming interfaces (API) or Web APIs that are accessed via Hypertext Transfer Protocol (HTTP)."
- Basically, web services allows different applications to talk to each other without human intervention required.
- There are two types of web services: REST (REpresentational State Transfer) and SOAP (Simple Object Access Protocol)
- The "Simple" part of SOAP is actually a misnomer as REST services are actually simpler than SOAP services.

The Basics: REST

→ AREST service is called using a standard URI (Uniformed Resource Identifier) like the following:

http://www.mycompany.com/axiom/getAccountCustomer?customerNumber=6

→ The format for the response of a REST service has no standard. It can be simple text, formatted as JSON (JavaScript Object Notation), or custom. It is up to the calling application to understand how the response will be formatted. In the example above, the response could look like either of the following:

JSON:

```
{
        "customer": {
                "companyCode": "RM",
                "customerNumber": "6",
                "name": "ENGINEERING EXCELLENCE, INC"
        }
}
```

Pipe Delimited:

"RM"|"6"|"ENGINEERING EXCELLENCE, INC"

The Basics: SOAP

A SOAP service uses XML for both the request and the response. A sample XML request would look like the following:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mes="http://wynnesystems.com/rm/schema/message"
xmlns:type="http://wynnesystems.com/rm/schema/type">
   <soapenv:Header/>
   <soapenv:Body>
      <mes:GetAccountCustomerRequest>
         <mes:filters>
            <type:filter>
               <type:fieldName>customerNumber</type:fieldName>
               <type:fieldValue>6</type:fieldValue>
               <type:operator>eq</type:operator>
            </type:filter>
         </mes:filters>
      </mes:GetAccountCustomerRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

The response would look like the following:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns2:GetAccountCustomerResponse
xmlns:ns2="http://wynnesystems.com/rm/schema/message"
xmlns:ns3="http://wynnesystems.com/rm/schema/type">
         <ns2:hasMore>false</ns2:hasMore>
         <ns2:totalRecords>1</ns2:totalRecords>
         <ns2:customers>
            <ns3:companyCode>RM</ns3:companyCode>
            <ns3:customerNumber>6</ns3:customerNumber>
            <ns3:name1>ENGINEERING EXCELLENCE, INC</ns3:name1>
         </ns2:customers>
      </ns2:GetAccountCustomerResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

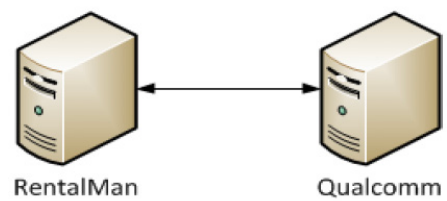
The Basics: REST vs SOAP

REST	SOAP
Assumes point-to-point communication model—not appropriate for distributed computing environment where message may go through one or more intermediaries	Designed to handle distributed computing environments
Minimal tooling/middleware is necessary. Only HTTP support is required	Usually requires tooling/middleware support
Formal description standards not in widespread use	Well defined mechanism for describing the interface. (XML + WSDL + XSD)
More verbose	Less verbose

The services offered in RentalMan are mainly SOAP services. They are aimed for communication between applications through middleware.

The History: Point to Point

- Prior to Integration Architecture, all integrations to RentalMan were point to point.
- An example was the integration between Qualcomm and RentalMan. A program read information from Qualcomm's web service and updated RentalMan tables and vice versa.



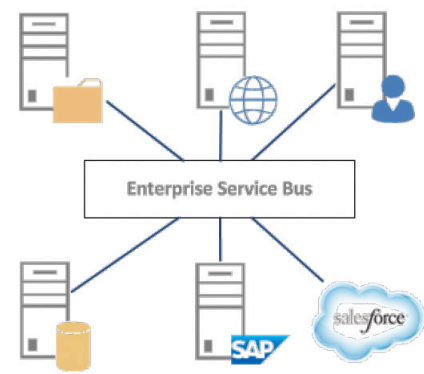
- Disadvantages to this approach:
  - **Fragile** – If either the web service or RentalMan tables changed, programming was required to fix it.
  - **Not reusable** – Programming changes would have to be done to integrate with different GPS vendors.

The History: Point to Point

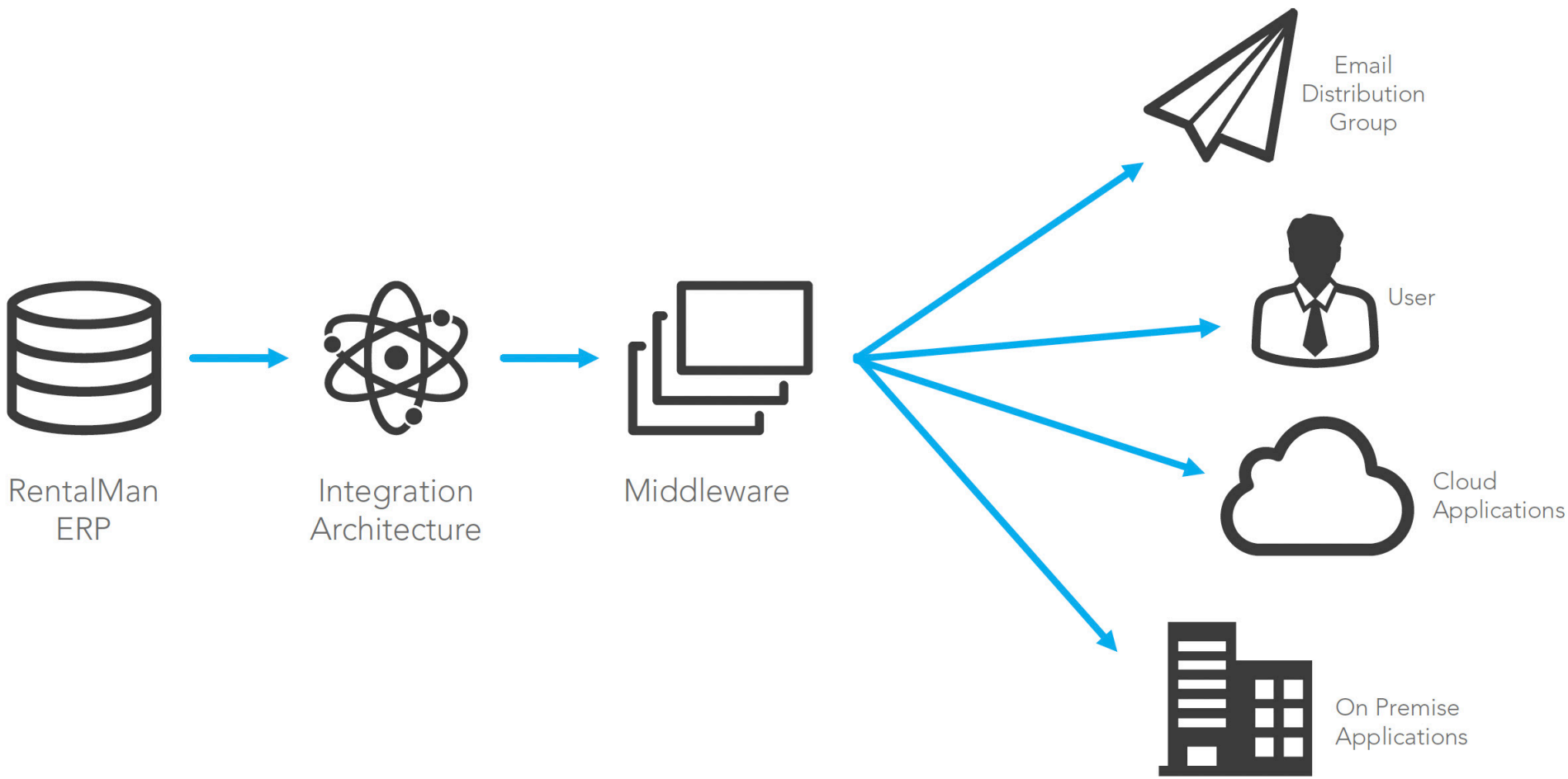
- To solve the shortcomings of point to point integrations, RentalMan Integration Architecture was created. RentalMan IA is a collection of web services that allow other applications to interact with RentalMan in a safe, robust manner.
- RentalMan IA exposes commonly used data and functions. Some examples are:
  - **GetAccountCustomer** – retrieves information about charge customers
  - **CreateAccountCustomer** – adds a charge customer
  - **GetUser** – retrieves information about a RentalMan user
  - **UpdateServiceMeterAndLocation** – Updates GPS data in RentalMan
  - …and many more

Middleware

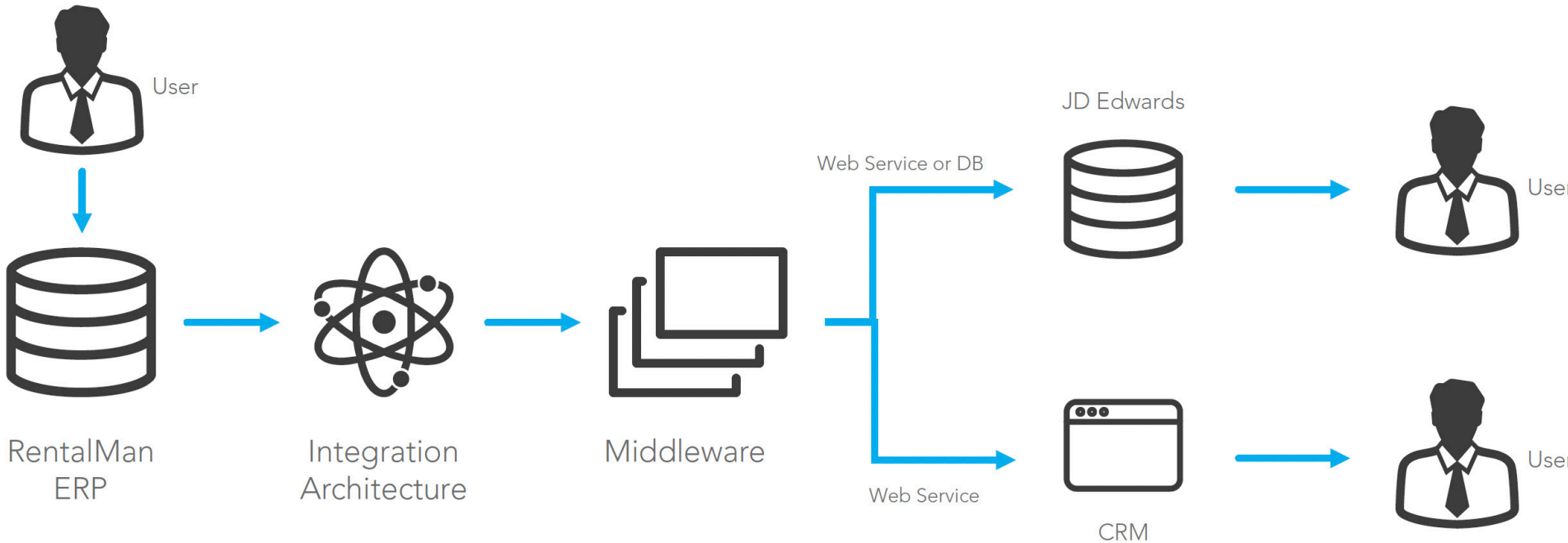
- Middleware will typically need to be employed to communicate between IA and external applications.
- An ESB (Enterprise Service Bus) is one type of middleware that facilitates communication between two disparate systems. This decouples systems from each other, allowing them to communicate without having a dependency or knowledge of other each other.
- Some examples of ESB products are:
  - Oracle ESB
  - Mule ESB
  - Jitterbit



How it Looks

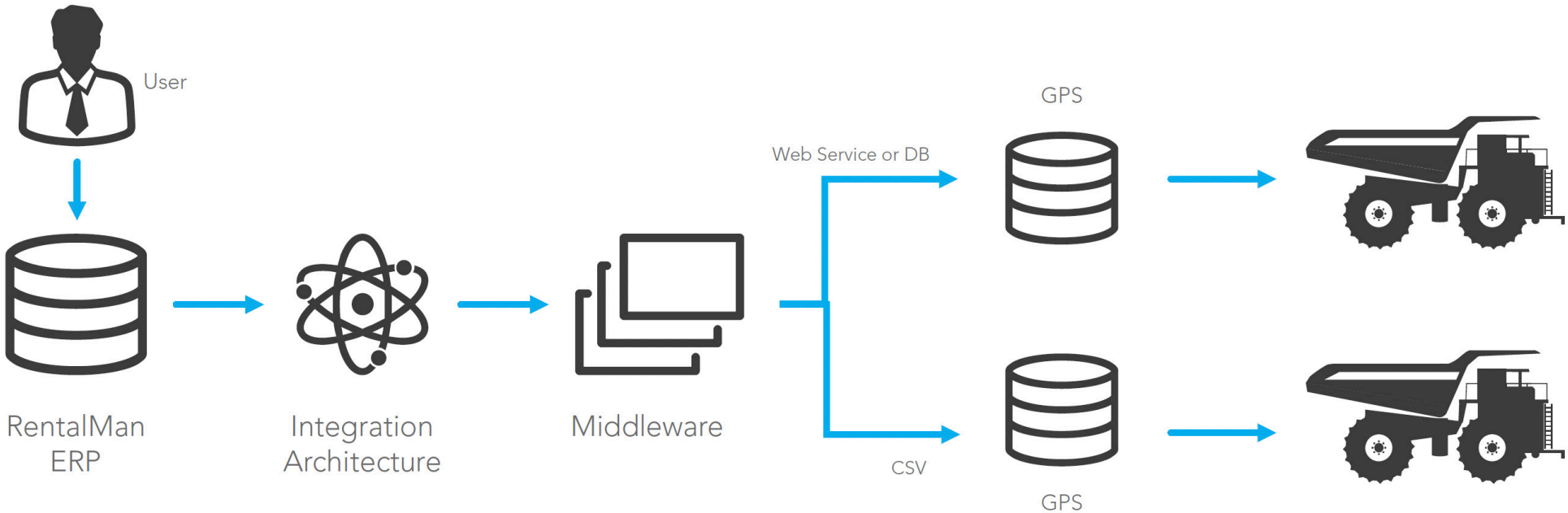


Example: Customers



- A rental company's parent corporation uses JD Edwards to maintain customers. They would like all customers created in JD Edwards to also be created in RentalMan. The middleware would be configured to retrieve any new JD Edwards customers either through the database directly or through a web service. It would then transform the data and send it to RentalMan through the CreateAccountCustomerweb service.
- The rental company uses a CRM for their inquiry to order process. Once a prospect is ready to place an order, the account needs to be set up in RentalMan. The middleware would be configured to retrieve the customer information through the CRM's web service and call IA's CreateAccountCustomerweb service.

Example: Telematics



- GPS provider #1 might have a web service called GetTerminalData. IA has a web service called UpdateServiceMeterAndLocation. The middleware will be configured to call GetTerminalData, transform the response, and call UpdateServiceMeterAndLocation. The middleware can be configured to run at any interval (every hour, minutes, second, etc).
- GPS provider #2 might not have any web services. Instead, on a nightly basis, it will create a comma separated values (csv) file and put it on a network folder. The middleware can be configured to monitor the network folder for any new files. Once it finds a new one, it will parse the csv file and send the information to UpdateServiceMeterAndLocation.

Wynne Systems develops and maintains a fully integrated ERP solution for rental equipment companies spanning the globe. With applications designed to help increase efficiencies across your entire organization, our comprehensive software suite gives you the tools to manage all aspects of your business with swift precision. We use the latest technologies to give you the competitive edge that will increase revenue, decrease costs and grow your operation.

Wynne Systems, Inc.
2603 Main Street
Suite 710
Irvine, CA 92614
marketing@wynnesystems.com
+1 949 224 6310