## **✅ RTBI – Knowledge Transfer Summary**

### **🔹 Overview**

RTBI (Right Time Business Intelligence) is a real-time train tracking and visualization tool used to monitor locomotive positions, fuel levels, and system activity using GPS and sensor data. It integrates multiple backend components including Oracle databases, brokers, FME servers, and Java-based services.

* **RTBI (Right Time Business Intelligence)** is a real-time, event-driven application used primarily in train tracking and management.
* It is a **single-page Angular application** supported by **NGRX (Redux-style state management)** and **Mapbox** for geospatial data.
* The app provides **real-time updates on trains, locomotives, air cars, and notifications** via a combination of **push and pull mechanisms**.

### **🔹 Key Components**

* **RTBI UI**: Displays live train data, assignments, and statuses.
* **EMPS**: Backend producer system that sends data to RTBI and other consumers.
* **FME (Field Messaging Engine)**: Dispatches messages to RTBI clients.
* **MQ/Brokers**: Middleware for message delivery.
* **Oracle DB**: Stores all backend data (Dev, UAT, Staging, Prod environments).
* **Map Viewer**: Supports geographic visualizations in RTBI.

### **🔹 Tools Used**

| **Tool** | **Purpose** |
| --- | --- |
| **Service Monitor** | Check real-time message flow to UI |
| **MQ Explorer / MQ Visual Edit** | Monitor queues and broker status |
| **SQL Developer** | Query Oracle databases (read mode) |
| **ServiceNow** | Incident and change management |
| **SharePoint** | Document repository (functional/technical) |

### **🔹 Common Incidents & Support Responsibilities**

* **User-specific vs. widespread issues**: Individual user problems are low priority; widespread ones require root cause analysis.
* **Incident Handling**:  
  + Monitor using ServiceNow.
  + Escalate to ITCC (Command Center) if thresholds are breached.
  + Identify issues across queues, DB, FME, and Java components.
* **Queue Management**:  
  + Monitor depth and lag.
  + Address "poison messages" causing loops.
  + Bounce apps or clean queues if needed.
* **Train Not Updating**:  
  + Verify GPS and message flow from White Tonics.
  + Analyze backend logs and UI latency.

### **🔹 Access & Environments**

* **Oracle DB Access**:  
  + Raise privileged access via ServiceNow.
  + Typically: Read/Write for Dev & UAT; Read-only for Prod.
* **Environments**:  
  + **Dev** – Development only.
  + **UAT** – User testing.
  + **Staging** – Pre-production testing with clusters.
  + **Prod** – Live environment with real-time data.

### **🔹 Functional and Defect Management**

* **Enhancements vs. Defects**:  
  + Enhancements = new features.
  + Defects = broken or misbehaving functionality.
* **Lifecycle**:  
  + Draft → Scoping → In Progress → Testing → Ready → Closed.

### **🔹 SharePoint & Documentation**

* **SharePoint URL**: Central repository for architecture, feature design, and troubleshooting documents.
* **How to use**: Search by keyword (e.g., “Train Low Fueled”) to locate relevant material.

### **🔹 KT Phases**

1. **Foundational Learning** – Tooling, environments, workflows, stakeholders.
2. **Discover & Simulate** – Deep dive into backend, logs, debugging scenarios.
3. **Demonstrate** – Live walk-throughs, real issue resolution simulations.

### **🔹 Required Access (L2 Support)**

| **Area** | **Access Needed** |
| --- | --- |
| Oracle DB | SQL Developer (Read mode) |
| Queues | MQ Explorer / MQ Visual Edit |
| UI Monitoring | RTBI Service Monitor |
| Incidents | ServiceNow |
| Docs | SharePoint |

### **🔹 Stakeholders (To Be Notified on Downtime)**

* **ITCC (Infra Command Center)** – Major outages
* **DBA Team** – Database issues
* **Java Services Team** – Middleware issues
* **MQ/Broker Admins** – Queue/dead-letter issues
* **Application Owners** – For RTBI/EMPS-specific topics

### **👥 Teams & Responsibilities**

* **Offering Manager**: Anderson
* **Support Team (L2)**: Zoltan, Willie, and team
* **Business**: Amy
* **UI/Frontend Development**: Rahul and team (L3)

### **🏗️ Architecture Overview**

* **Frontend**: Angular 14, Material, Google Charts, Mapbox
* **State Management**: NGRX (actions → effects → reducers → selectors → facades)
* **Push Mechanism**: FME Server (messages to UI clients)
* **Pull Mechanism**: Web service calls during initial load and on-demand refreshes
* **Polling Mechanism**: Used specifically for notes (every 30 seconds)
* **Back End**: EMPS services (data source for trains, notifications, etc.), WebSphere-hosted Java services

### **⚙️ Key Features**

1. **Trains, Locomotives, Air Cars**
   * Data retrieved via EMPS services
   * Maintained in NGRX feature slices
2. **Notifications**
   * Informative, persistent messages
   * Displayed in bottom panel
   * Always visible if applicable
3. **Alerts**
   * Critical and non-critical (e.g., brake failures)
   * Cause banners to appear
   * Can be manually dismissed (unlike notifications)
4. **Bookmarks**
   * User-defined saved views (zoom, notification subscriptions, etc.)
   * Stored in **Azure PostgreSQL** now (previously in Oracle)
   * Applied on initial load or manually
5. **Disaster Recovery**
   * Simulated 2x/year
   * App is expected to be up within 8 hours
   * Active-active mirror setup

### **📉 Performance & Monitoring**

* **Dynatrace** is used to monitor:  
  + Memory/CPU usage
  + UI slowness
  + WebSphere service health
* **Azure Explorer** helps debug user-specific issues (status codes, logs)

### **🛠️ Support Details**

* **L1**: ITCC (IT Command Center) receives alerts
* **L2**: RTBI support team investigates and coordinates fixes
* **L3**: Development team (UI/backend)
* **French language support** is available for Quebec users
* **Alerting** via ITCC: on server/app/queue issues

### **📨 Email Notifications**

* Enabled via a separate admin tool (not in RTBI UI)
* Triggered through ANSE (notification as a service)

### **🔜 Upcoming Enhancements**

* **Architecture revamp**: Replace ODM & complex event processor with **Databricks**, **Kafka**, and **microservices**
* **Upgrade to Angular >14** planned
* Move from polling to push for notes (planned)

### **🧾 Miscellaneous**

* No Firebase used
* Active Directory used for user roles/security groups (especially under Rule 112)
* UI logs stored per user; backend logs on WebSphere

### **✅ Important Takeaways**

* Understand **NGRX data flow**
* Be aware of **push vs pull vs polling**
* Know how to monitor and debug issues using **Dynatrace** and **Azure Explorer**
* Be familiar with **bookmarks, alerts, notifications**
* RTBI heavily depends on **EMPS backend** for data