**Data Integration Capabilities**

WSO2 provides several capabilities to integrate and manage data effectively. Here’s a breakdown of these capabilities and their purpose:

* **Expose Data as SOAP/REST**: Allows you to make data accessible as web services, either using SOAP (Simple Object Access Protocol) or REST (Representational State Transfer) APIs. This makes data available for various applications, regardless of platform or technology.
  + **Example**: A retail company wants to expose its product catalog as a REST API to enable different e-commerce platforms to access product details.
* **Support Datasources OOTB (Out of the Box)**: WSO2 supports multiple standard datasources, such as MySQL, Oracle, and PostgreSQL, directly without custom configurations.
  + **Example**: Connecting a MySQL database that contains customer records and exposing it for a customer service application.
* **Support Custom Datasources**: In addition to standard databases, WSO2 lets you connect to other custom or non-standard datasources by configuring them.
  + **Example**: Integrating with a proprietary database or a non-relational datasource used by a legacy system in a hospital.
* **Batch Requests**: Supports the handling of multiple requests as a single batch, improving efficiency and reducing the number of network calls.
  + **Example**: Processing multiple orders at once instead of handling each order request individually.
* **Data Federation**: Combines data from multiple sources to present a unified view, without moving the data from its original locations.
  + **Example**: A bank combining data from customer accounts in different branches into a single, consolidated report.
* **Transactional Data Access**: Ensures that data operations are consistent and reliable by supporting transactions (e.g., commit and rollback).
  + **Example**: A payment system where funds are debited from one account and credited to another; if one step fails, the transaction rolls back to maintain consistency.
* **Transform and Validate Data**: Enables data transformation and validation to ensure data consistency and correct format before processing.
  + **Example**: Converting currency values to a common unit or validating input fields for a user registration service.
* **Secured and Managed Data Access**: Provides secure data access by managing permissions, authentication, and encryption.
  + **Example**: Restricting access to sensitive medical records only to authorized healthcare professionals.

**2. Creating a Data Service**

Creating a data service in WSO2 involves defining a service that allows other applications to access and manipulate data. Here’s the general workflow:

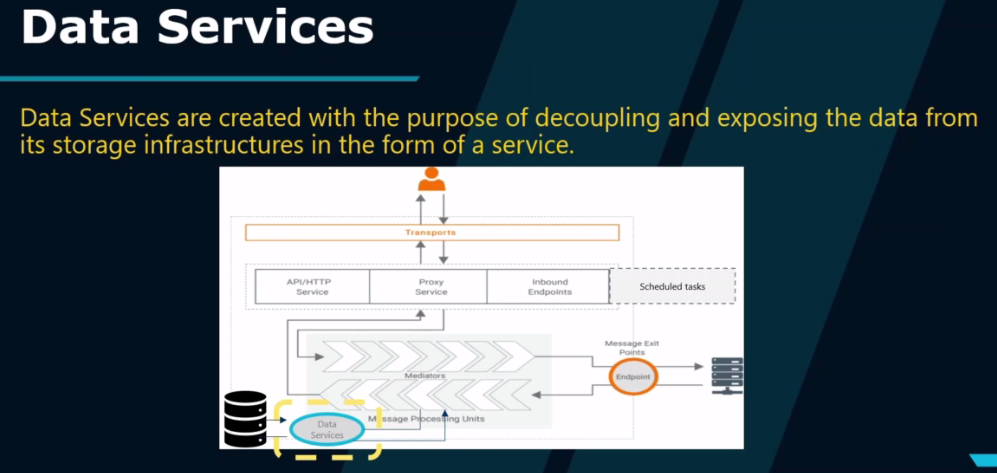
* **Define Data Service**: Set up the data service and give it a name.
* **Add Datasource**: Connect to the datasource (e.g., a database).
* **Add Query**: Define the SQL queries or data-fetching methods for your service.
* **Expose as SOAP or REST**: Finally, make the service available as a SOAP or REST API.

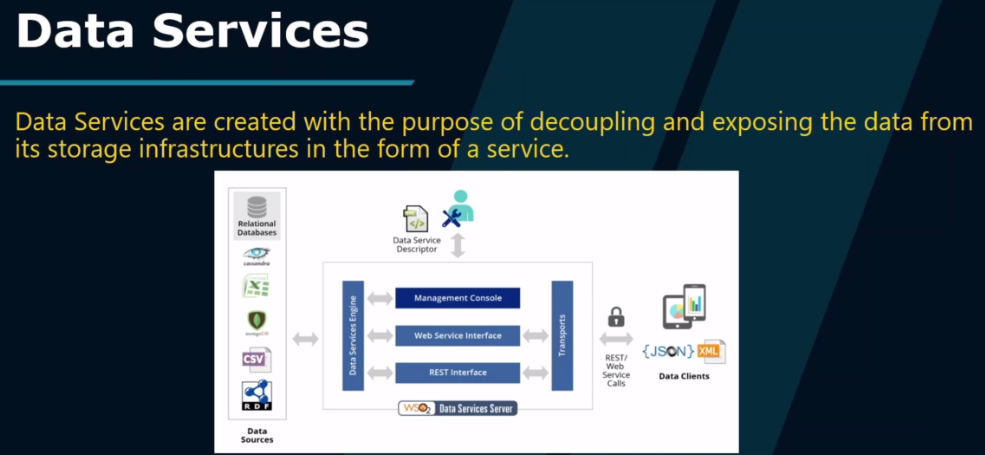
**Real-World Example**

Let’s say a company wants to create a data service to access employee records stored in an HR database. The steps would be:

1. **Define Data Service**: Name the service “EmployeeService.”
2. **Add Datasource**: Connect it to the HR database.
3. **Add Query**: Define queries like “GetEmployeeDetails” to fetch employee data by ID.
4. **Expose as REST**: Make this service accessible as a REST API, allowing applications within the company to retrieve employee information using endpoints.

This data service allows secure, standardized access to employee data, making it easy to integrate with other systems (e.g., payroll or attendance systems).





The image explains how **Data Services** work in WSO2. Here’s a simple breakdown:

1. **Data Sources**: Various types of data sources (e.g., relational databases, Cassandra, CSV files) hold the actual data.
2. **Data Service Engine**: The core engine in WSO2 reads data from these sources using a **Data Service Descriptor** (which defines what data to expose and how).
3. **Interfaces**:
   * **Web Service Interface** and **REST Interface**: These interfaces expose the data as either SOAP or REST services, making it accessible to clients.
4. **Management Console**: A control panel where you can manage and configure data services.
5. **Data Clients**: Applications or systems that use the data service by sending requests (REST or SOAP) and receive responses in **JSON** or **XML** format.

**How It Works Together**

The Data Service Engine connects to data sources, retrieves the required data, and sends it to clients in a standardized format (JSON or XML) via REST or SOAP, allowing easy and secure data access for applications. This decouples data from its storage and makes it available as a service.