**Oracle Data Integrator 12c: Integration and Administration**

**🔧 Core Concepts Introduced**

**1. ETL vs. ELT**

* **Traditional ETL**: Extract → Transform → Load  
  Staging area on a **separate server**.
* **ODI's ELT**: Extract → Load → Transform  
  Staging area is on the **target server** → better **performance**.

**2. Change Data Capture (CDC)**

* Only loads **changed data** (inserted/updated/deleted).
* Improves performance over full dataset loads.

**3. Declarative Design**

* ODI automates SQL/script generation based on user-defined mappings.
* Enhances developer productivity.
* **Knowledge Modules**:
  + Code templates used to perform operations.
  + Can include SQL, OS commands, scripts.

**🛠️ ODI Architecture & Components**

**ODI Studio**

* Main GUI to manage and develop:
  + Designer Navigator → Mappings, procedures, packages, load plans.
  + Topology Navigator → define data sources/targets.
  + Operator Navigator → monitor executions.
  + Security Navigator → manage access control.

**ODI Agents**

* Execute and schedule ODI jobs (discussed later).

**Repositories**

* **Master Repository**:
  + Stores topology and security-related metadata.
* **Work Repository**:
  + Stores mappings, procedures, execution logs.

**💻 ODI Studio Setup**

* You connect to repositories via **JDBC**.
* Required details:
  + Host (e.g., localhost)
  + Port
  + Instance ID (e.g., ORCL)
* User: SUPERVISOR (default admin account).
* Wallet setup is optional, used to save login credentials.

**📂 Navigators in ODI Studio**

* **Designer**: Create and manage mappings, procedures, variables.
* **Topology**: Define data servers and physical/logical architecture.
* **Operator**: View execution results.
* **Security**: Manage user roles and permissions.

🧠 **Repository Storage Summary**

| **Object Type** | **Stored In** |
| --- | --- |
| Mappings, Packages | Work Repository |
| Execution Logs | Work Repository |
| Topology Objects | Master Repository |
| Security Settings | Master Repository |

**🔧 ODI Agent – The Runtime Brain of ODI**

* **ODI Agent** is a lightweight Java program that **runs the integration process**.
* At **runtime**, it:
  + Connects to the **repository**
  + Picks up **Knowledge Modules** and **Mapping info**
  + **Generates and runs code** on your **source & target systems**
  + **Updates session logs** with the result (success/failure)

**🔁 Agent Execution Modes**

1. **On-demand execution**: Manual runs of mappings/load plans
2. **Scheduled execution**: Automated via **schedules**

**👨‍🔧 Types of ODI Agents**

1. **Standalone Agent**:
   * Runs in its **own JVM** (independent)
   * Can be installed anywhere with Java
2. **Colocated Standalone Agent**:
   * A **standalone agent** installed **alongside** an app server (like WebLogic)
   * Shares some libraries but otherwise independent
3. **Java EE Agent**:
   * Fully **integrated with your application server** (e.g., WebLogic)
   * Supports:
     + **Clustering**
     + **High availability**
     + **Connection pooling**
     + **Load balancing**

**🌐 ODI Console**

* A **web-based** tool to **view and manage** execution (mappings, load plans, etc.)
* Cannot **edit** objects
* Ideal for **operations teams** to monitor jobs

**🔍 Enterprise Manager (OEM) Integration**

* ODI can plug into **Oracle Enterprise Manager (OEM)**
* Through **ODI Management Pack**, you can:
  + Monitor all components: agents, repos, source/target DBs
  + Track config changes
  + Drill into session logs
  + Manage SLAs and performance

**📦 ODI Repositories**

There are two main types:

1. **Master Repository**
   * Stores:
     + **Security data**
     + **Topology info**
     + **Version control**
   * Each **Work Repository** can only be attached to **one** Master Repo
2. **Work Repository**
   * Stores:
     + **Models** (tables/columns)
     + **Projects** (mappings, packages, procedures)
     + **Session logs**
   * Two types:
     + **Development Work Repository**: Full access (create models, mappings, etc.)
     + **Execution Work Repository**: **Only run** & monitor (no editing)

**🏗️ Repository Deployment Scenarios**

1. **Single Master Repo with Dev, QA, and Prod Work Repos**
   * Dev & QA: Full access
   * Prod: Execution-only
2. **Separate Repos across Firewall**
   * One Master for Dev/QA
   * A **different Master** for Prod (isolated setup)

**🧪 Goal of Lab 1**

This lab introduces:

* The **ODI Studio interface**
* How to **reset** the lab environment using a command
* How to **connect to the ODI repository**

**🖥️ 1. Exploring the Desktop**

You are introduced to 3 main icons on the desktop:

* ✅ **ODI Studio**: The main tool you'll use for development.
* 🛠️ **SQL Developer**: Used later for database interactions.
* 💻 **Terminal**: Used to run scripts via command line.

**🔁 2. Resetting the Lab Environment**

You start by:

* Opening **Terminal**
* Navigating to the lab folder using cd command
* Running the script: reset01

🧠 **Why this is done**:

It **copies a pre-configured repository** into ODI Studio so you're ready for hands-on work.

⚠️ Important: Make sure you're **not connected to any repository in ODI Studio** before running the script.

**🚀 3. Starting ODI Studio**

* Double-click on the **ODI Studio icon**.
* Once it opens, you'll land on the **Start Page**.
* You’ll see these 3 main navigators on the left:
  + 👷 **Designer** – where you create and manage ETL processes.
  + 🌐 **Topology** – where you define technologies, data servers, and physical/logical architecture.
  + 👀 **Operator** – for viewing logs and monitoring processes.
* You can also enable **Security Navigator** from the Window menu.

**📘 4. Using the Built-in Help**

* The **Start Page** offers access to tutorials and documentation.
* There's also a **Help** menu with:
  + Table of Contents
  + Searchable topics (like Load Plans, Temporary Objects)
* You can choose to **hide these on startup** by unchecking a box.

**🔗 5. Connecting to the Repository**

You're connecting to the repository that was imported via the reset script:

1. Click **Connect to Repository**.
2. The **Login screen** shows up. Click the **green plus (+)** to create a new connection.
3. Fill in these fields:

| **Field** | **Description** |
| --- | --- |
| **Login Name** | Any name you choose (e.g., ODI\_REPO) |
| **Username** | SUPERVISOR (default admin) |
| **Password** | From activity guide (e.g., welcome1) |
| **DB Connection (JDBC URL)** | Format: jdbc:oracle:thin:@localhost:1521:ORCL |
| **Work Repository** | Select the one shown (fetched from the DB) |

1. Test the connection – it should say ✅ **Success**.
2. Click **OK**.

**🔐 6. Wallet Password (Optional)**

This step lets you **store login credentials** so you don't have to enter them every time:

* Choose to store them **in a wallet**.
* Set a **wallet password** (e.g., welcome1).
* You can set it to never expire, or just use defaults.

**🧭 7. Navigating the Repository**

Now you're logged into ODI Studio and connected:

* Go to the **Designer Navigator** tab.
* Expand the HandsOn folder to see:
  + **Packages**
  + **Mappings**
* These will be used in future labs.

**🔚 8. Closing the Lab**

* To exit: File > Exit
* This will close ODI Studio and disconnect all sessions.

**🧠 Lesson 2 Summary: Administering ODI Repositories**

**🔹 What Are We Learning?**

* How to **create and manage** the **Master Repository** and **Work Repository** in **Oracle Data Integrator (ODI)**.
* Use of **Repository Creation Utility (RCU)** for setting up repositories.
* How to **connect**, **export**, and **import** these repositories.

**📁 1. What Are ODI Repositories?**

ODI has two types of repositories:

* **Master Repository** – stores global information (security, topology).
* **Work Repository** – stores project-related data (mappings, procedures).

Both are stored **in a database**, and you’ll need:

* A **user with DBA or SYSDBA privileges**
* A **schema**, **tablespaces**, and correct **permissions**

**⚙️ 2. How to Create Repositories Using RCU**

**🧰 Step-by-step:**

1. **Open Terminal**
   * Navigate to:

bash

$FMW\_HOME/oracle\_common/bin

* + Run: rcu

1. **RCU Wizard**
   * Choose: **"Create Repository"**
   * Optionally: Prepare scripts for DBA if you can’t run them yourself.
   * Choose **“Create and Load”** if you have full access.
2. **Database Connection**
   * Host: localhost
   * Port: 1521 (default for Oracle)
   * Service: oracle.us.oracle.com
   * User: sys as sysdba
   * Enter SYS password
3. **Component Selection**
   * Prefix (e.g., DEV1)
   * Select: **Oracle Data Integrator** → this creates:
     + Master and Work Repository
4. **Passwords**
   * Use **same password** for all schemas (easier for development)
5. **Work Repository Setup**
   * Supervisor user is always created for Master Repo
   * Choose: **Development Repository** (for full access)
   * Name: e.g., WorkRep1
   * Set password and confirm
   * Choose encryption: **AES-128** (default)
6. **Tablespace**
   * Default: tables and temp spaces
   * RCU (Repository Creation Utility) will **create them if missing**
7. **Review Summary**
   * Check all details and click **Create**

**💻 3. Connecting to Repositories in ODI Studio**

**🔐 Steps:**

1. Open **ODI Studio**
2. Click **“Connect to Repository”**
3. If prompted, enter **wallet password**
4. Click **New Connection**
   * Login name: dev1\_ODI\_repo
   * Supervisor password
   * Use **JDBC connection**
   * Provide:
     + Driver
     + URL (JDBC string with host, port, SID)
5. Choose your **Work Repository**
6. Click **Test → OK → Connect**

**🗂 4. What Can You Do After Setup?**

* Navigate to **Topology Navigator**
  + View **Master** and **Work Repositories**
  + **Change passwords**
  + **Add new Work Repositories**
* Go to **Topology → Export**
  + Export Master or Work Repository
  + Similarly, you can **import** them

🧩 Easy Definitions

| **Term** | **Meaning** |
| --- | --- |
| **RCU** | Tool to create and manage ODI repositories |
| **Supervisor** | Default admin user for ODI Master Repository |
| **JDBC** | A way to connect Java apps (like ODI) to a database |
| **Wallet** | Secure way to manage passwords |

**✅ Key Concepts of ODI Topology – Lesson 3 Summary**

**🔸 1. What is Topology in ODI?**

* Topology = A **representation of your information systems** inside ODI.
* Connects **source** and **target** systems via **integration processes**.
* Used to define and manage **infrastructure, connections, and environments**.

**🔸 2. Core Components of ODI Topology**

**📌 A. Physical Architecture**

* Defines the **actual physical resources** (servers, files, DBs).
* Includes:
  + **Technologies** (Oracle, SQL Server, DB2, Hive, etc.)
  + **Data Servers** (e.g., Oracle instance, SQL Server instance, File Server)
  + **Physical Schemas**
    - Subdivisions within a data server.
    - Made of:
      * **Data Schema** – stores actual data.
      * **Work Schema** – stores temporary objects (e.g., staging tables).

**📌 B. Logical Architecture**

* Used at **design time** to define mappings, packages, and procedures.
* Independent of physical storage location.
* Includes:
  + **Logical schemas:** Aliases or placeholders for the real schemas.
  + **Logical Agents**: Represent the programs that run your data jobs.

**📌 C. Contexts**

* **Bridge between logical and physical layers**.
* Determine **which physical schema** a logical schema maps to **at runtime**.
* Can be **geographical** (e.g., Boston, Tokyo) or **environmental** (Dev, Test, Prod).
* Mapping: One logical schema ➝ One physical schema **per context**.

**🔸 3. Important Guidelines & Recommendations**

1. **Each physical data server must be defined once** in the topology.
2. **Each subdivision** (e.g., schema, DB) within a data server must be defined as a **physical schema**.
3. **All schemas** within the **same DB instance** should be defined under **one data server**.
4. Always define a **default schema** for each data server (needed for knowledge modules).
5. **Logical schemas** should be created for **each group of physical schemas** that share the **same structure**.
6. A **context must be defined** for each runtime environment or physical location.

**🔸** **4.** **Schema Naming Conventions by Technology**

| **Technology** | **Data Server** | **Schema Equivalent** |
| --- | --- | --- |
| Oracle | Instance | Schema |
| SQL Server/Sybase | Server | Database/Owner |
| DB2 | Server | Library |
| Teradata | Server | Schema |
| MS Access | File | No schema |
| JMS Topics | Router | Topic |
| Flat Files | File Server | Directory |

**🔸 5. Examples & Real-world Scenario**

**🧩 Boston Production Site**

* **Windows Machine**
  + SQL Server instance ➝ 1 Data Server with 2 Physical Schemas (DW & Purchasing)
* **Linux Machine**
  + Oracle 10g ➝ 1 Data Server ➝ Accounting Schema
  + Oracle 12c ➝ 1 Data Server ➝ Sales Schema

**🧩 Tokyo Production Site**

* 2 Windows Machines ➝ 2 SQL Server instances ➝ Separate Data Servers for DW & Purchasing
* 1 Linux Machine ➝ 1 Oracle instance ➝ 1 Data Server with 2 Schemas (Accounting & Sales)

**🧩 New York Development Site**

* 1 Windows Machine ➝ 1 Oracle + 1 SQL Server instance
  + Oracle ➝ Accounting & Sales
  + SQL Server ➝ DW & Purchasing

📝 **Note:** It’s **not the physical machine** but the **number of DB instances** that determines how many data servers to register.

**🔸 6. Logical Schema Mapping via Contexts**

* **Logical schemas** abstract the underlying physical schemas.
* At runtime, the **selected context** determines which physical schema is used.
* E.g., Logical schema accounting can map to:
  + Boston accounting schema via Boston context.
  + Tokyo accounting schema via Tokyo context.
  + New York accounting schema via NY context.

**🔸 6. Key Terms Clarified**

| **Term** | **Meaning** |
| --- | --- |
| **Data Server** | System storing data (DB, file server, etc.) |
| **Physical Schema** | Subdivision within a data server; has actual/work schema |
| **Logical Schema** | Design-time abstraction of physical schemas |
| **Context** | Runtime mapper of logical to physical schema |
| **Work Schema** | Stores temporary objects like staging tables created during data integration |
| **Technology** | Type of data source (e.g., Oracle, SQL Server, File, JMS) |

**🧠 Lesson Summary: ODI Agents & Topology Configuration**

**🔹 ODI Agents: Purpose and Types**

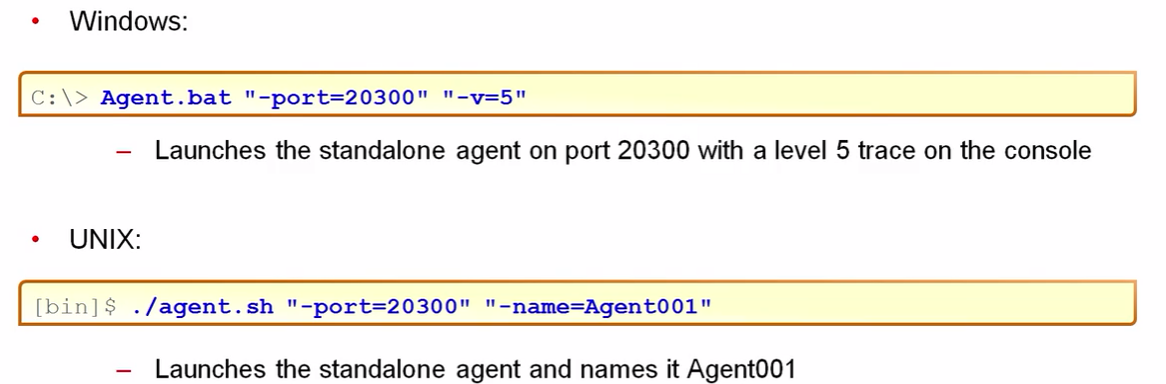
* **Agents** are lightweight Java runtime components responsible for **coordinating integration execution** in Oracle Data Integrator (ODI).
* They:
  + Connect to the **ODI repository**.
  + Pick up **mappings** and **knowledge modules**.
  + Generate and execute code on **data servers**.
  + Retrieve and update **session logs** with execution results.
* Can be invoked via **GUI, CLI, or schedules**.

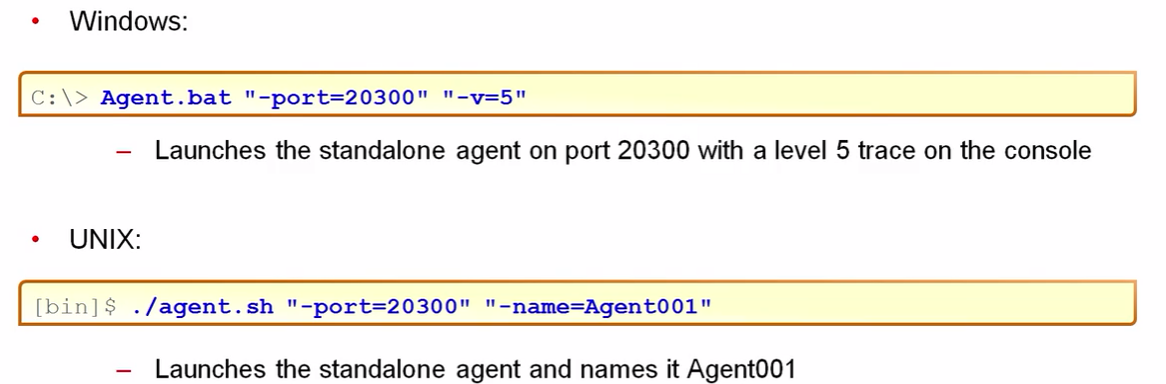
**🧩 Agent Types**

1. **Standalone Agent**:
   * Simple JVM-based.
   * Can run on any machine (source, target, or third server).
2. **Colocated Standalone Agent**:
   * Runs in conjunction with **WebLogic Server** (WLS).
   * Shares some WLS libraries but is otherwise independent.
3. **Java EE Agent** (used in **production**):
   * Deployed as a **web application**.
   * Leverages **WebLogic** for advanced management.

**🛠️ Creating a Physical Agent: Step-by-Step**

1. **Navigate to the Configuration Wizard**:
   * Path: Fusion Middleware Home > oracle\_common > common > bin
     + cd FMW\_HOME/oracle\_common/bin
   * Run: config.sh (on Unix/Linux)
2. **Create a New Domain**:
   * Replace base domain name with agent-specific name (e.g., Agent1).
3. **Template Selection**:
   * Use built-in templates (e.g., **Colocated Standalone Agent** template).
4. **Configure for Development or Production**:
   * Development: Easier setup (uses boot properties).
   * Production: More secure (manual credentials entry).
5. **Configure JDBC Settings**:
   * Provide DB host, service name (orcl.us.oracle.com), port (1521), schema owner, and password.
   * Click “Get RCU Configuration”.
6. **Link to ODI Repositories**:
   * Connect to **Master** and **Work** repositories.
   * Verify schema and JDBC connections.
7. **Agent Deployment Details**:
   * Choose **component type** (ODI) that you want to configure (Eg. System Component).
   * Set **listening port** and **supervisor credentials**.
8. **Start Agent**:
   * Navigate to: FMW\_HOME/user\_projects/domains/agent1/bin
   * Run: ./agent.sh -NAME=<agent\_name>

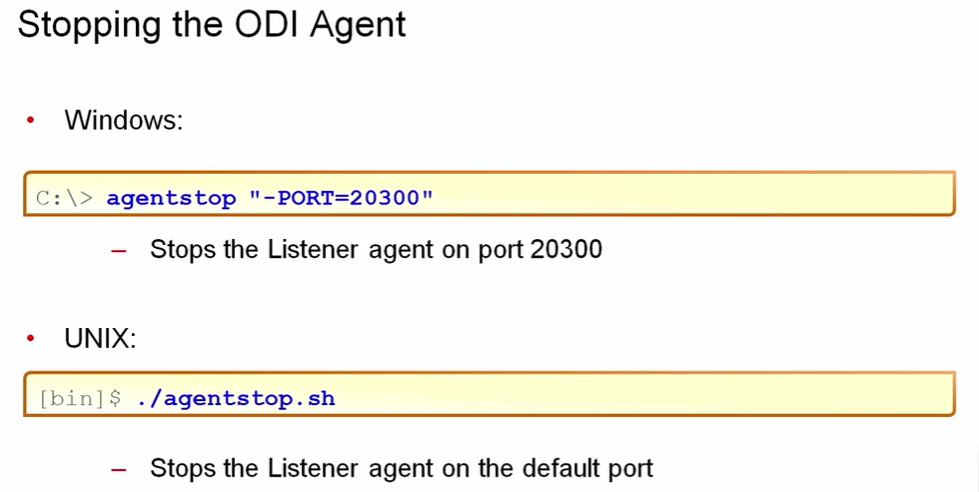


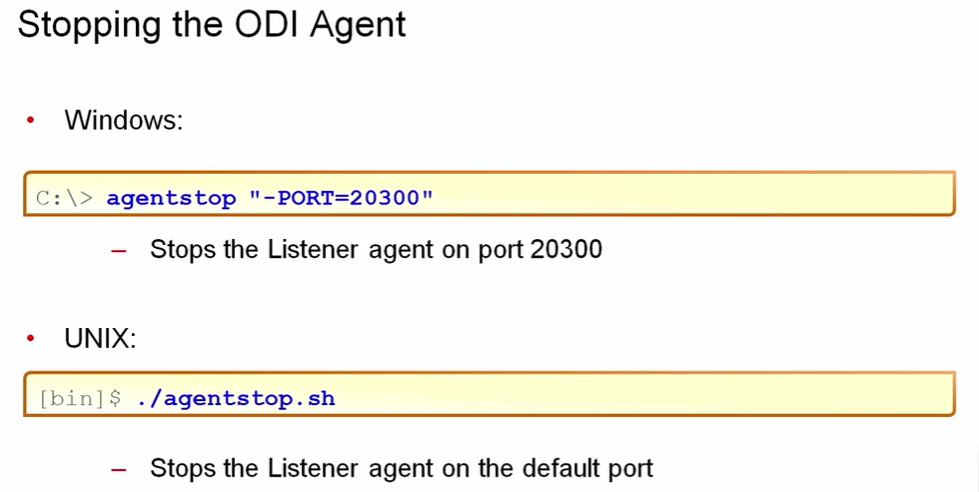


* + Run: startnodemanager.sh or startagent.sh

1. **Register Agent in ODI Studio**:
   * Go to **Topology tab** → Right-click Agents → **New Agent**
   * Enter name (OracleDIAgent1), host (localhost), port, and **click Save**.
   * Test connection to confirm the agent is active.

**Stopping the agent**





Stops the Listener agent on the default port

**🧱 Physical vs Logical Agents**

* **Physical Agent**: The actual runtime component.
* **Logical Agent**: An abstraction used in **contexts** to link physical agents.
* Each physical agent must be defined in ODI for each started instance.

**🔄 Load Balancing Between Agents**

* Configure agents to offload to others if heavily utilized.
* Example:
  + agent86 can offload to OracleDIAgent1
  + agent99 can offload to agent86 and OracleDIAgent1

**🧪 Development Best Practice: Mimic Fragmented Production**

* Even if development has fewer DB instances, **mimic fragmentation**:
  + Example: In production, Boston has 2 Oracle instances → Development should define 2 Oracle **data servers**, even if only one instance exists.
* Helps maintain environment consistency.

**🗂️ Topology Planning Recommendations**

1. **Identify all data servers – Every physical source or database.**
2. **List all physical schemas – One for each application and environment.**
3. **Decide physical agent placement – Choose where agents will run.**
4. **Identify contexts – Dev, Test, Prod, etc., based on combinations of agents and schemas.**
5. **Define logical architecture:**
   * **Logical schemas – Abstract names for physical schemas.**
   * **Logical agents – Abstract names for agents.**
6. Use a **matrix** to map contexts, logical schemas, and physical schemas.
   * **Rows** = Contexts (e.g., Development, Testing, Production)
   * **Columns** = Logical Schemas
   * **Cells** = Mapped Physical Schemas

**🧠 Quiz Questions Recap**

1. **If you want to use 1 subdivision of a data server, how many physical schemas to define?**
   * ✅ A: **1**
2. **Max number of physical resources a logical resource maps to in one context?**
   * ✅ B: **1**
3. **How many contexts can use a single physical schema?**
   * ✅ D: **Any number**

**ODI Topology – Data Servers, Schemas, Contexts & Agents**

**🔹 1. Data Servers**

**Definition: A data server represents a physical connection to a data source (e.g., Oracle, SQL Server, flat file, XML, etc.).**

**✅ Creation Steps:**

* **Go to Topology Navigator → Find the Technology (e.g., Oracle).**
* **Right-click → New Data Server.**
* **Enter:**
  + **Name, User, Password.**
  + **JDBC Driver (recommend Type 4) – allows direct TCP/IP access.**
    - **Type 3: Needs a middleware server (3-tier).**
    - **Type 2: Requires native DB client.**
    - **Type 1: Uses ODBC bridge.**
  + **JDBC URL – defines how to connect; may contain driver-specific parameters.**

**🔁 JNDI Alternative:**

* **Select the JNDI checkbox on the Definition tab.**
* **Go to the JNDI tab → Set parameters.**
* **Test using:**
  + **Local (No Agent) for dev**
  + **Or choose a running ODI Agent.**
* **Test validates:**
  + **Driver, URL, connectivity.**

**🔹 2. Physical Schema**

**Definition: Represents specific schemas inside a database/data server. A single data server can have multiple physical schemas.**

**✅ Steps to Create:**

* **Right-click on the Data Server → New Physical Schema.**
* **Don't click "New" on the Technology level (creates a new server by mistake).**
* **You’ll define:**
  + **Data Schema – where real tables are stored (e.g., SALES\_DEV).**
  + **Work Schema – used by ODI for temp/error/integration tables (e.g., ODI\_TEMP).**

**🏷️ Table Prefixes:**

* **Temporary and error tables:**
  + **C$ = Loading tables**
  + **I$ = Integration tables**
  + **E$ = Error tables**
* **These prefixes can be changed but are standardized and widely recognized – keep them unchanged if possible.**

**Example:**

* **Data Schema = SALES\_DEV**
* **Work Schema = ODI\_TEMP**
* **Multiple physical schemas (e.g., SALES\_PROD) can share a single work schema.**

**⚠️ Common Mistake:**

**Creating both DEV and PROD physical schemas on the same data server (done for demo, but not realistic in production).**

**🔹 3. Logical Schema**

**Definition: An abstract name used by ODI that maps to different physical schemas depending on the context (DEV, PROD, etc.).**

**✅ Steps to Create:**

* **Go to Logical Architecture → Technology (e.g., Oracle).**
* **Right-click → New Logical Schema.**
* **Enter:**
  + **A logical name (e.g., ORACLE\_ORCLLOCAL\_SALES).**
* **For each Context (DEV, PROD, GLOBAL), map to:**
  + **The relevant Physical Schema (e.g., SALES\_DEV, SALES\_PROD).**

**Example Mapping:**

| **Context** | **Physical Schema** |
| --- | --- |
| Development | ORACLE\_ORCLLOCAL.SALES\_DEV |
| Production | ORACLE\_ORCLLOCAL.SALES\_PROD |
| Global | ORACLE\_ORCLLOCAL.SALES\_DEV |

**⚠️ Important:**

* If you don’t define a context mapping:
  + No error at design time.
  + But you **cannot use the schema at runtime** under that context.

**🔹 4. Agents**

**Definition:** Agents are services that execute the ODI scenarios and load plans.

**✅ Logical Agent:**

* Go to **Logical Architecture** → Agents → Right-click → **New Logical Agent**.
* Enter a **name** (e.g., LocalAgent).

**✅ Context Mapping:**

* Open the **Contexts** section → Double-click a context (e.g., Development).
* Go to **Agents tab**:
  + Map **Logical Agent** → **Physical Agent**.

**Example:**

| **Context** | **Logical Agent** | **Physical Agent** |
| --- | --- | --- |
| Development | LocalAgent | OracleDIAgent1 |
| Production | StandaloneAgent | OracleDIAgent1 |

In production, these mappings would differ — in the demo only one physical agent is used for all contexts.

**🔹 5. Context**

**Definition:** Represents an environment like Development, QA, Production. Allows mapping of logical objects (schemas, agents) to different physical objects.

**✅ For Each Context:**

* You must map:
  + **Logical Schema → Physical Schema**
  + **Logical Agent → Physical Agent**

**Steps:**

* Go to **Topology Navigator → Contexts**.
* Open a context → Set mappings under **Schemas** and **Agents** tabs.

**✅ Final Summary of Key Concepts**

| **Concept** | **Description** |
| --- | --- |
| **Data Server** | Represents a physical data source (e.g., Oracle instance) |
| **Physical Schema** | Real schema inside DB server (Data + Work schemas) |
| **Logical Schema** | Abstraction that maps to different physical schemas per context |
| **Agent** | Executes jobs – can also be logically mapped |
| **Context** | Environment-specific mapping (DEV, QA, PROD) |