A Comprehensive Guide to the Traceability Driver

The Traceability Driver is a ready-to-use solution that streamlines integration between traceability systems with standardized data. Initially built for the seafood industry using the GDST standard, it is now expanding to support other commodities. It syncs and converts traceability data into standard formats automatically, allowing organizations to share data securely and affordably without major system changes.

By deploying the Traceability Driver on-premises or in your own cloud environment, you maintain full control over your data. The system only reads data from your existing database and does not write back to it, ensuring your original system remains secure. Data is transferred into a separate Data Cache, where it is stored and accessed by the prebuilt API. The system is configured with secure authentication methods such as OAuth or API keys, allowing you to control who has access to your traceability data. With this setup, all traceability information is kept within your organization's network, preserving both data integrity and confidentiality.

How Does the Traceability Driver Work?

The Traceability Driver operates by mapping data from an existing system into standardized events and master data. This data is then stored in a separate database called a Data Cache, which serves as the data source for API queries. The Traceability Driver works with SQL Server or MySQL internal databases and can be extended to support other database types by re-implementing the ITDConnector interface. The Data Cache supports SQL Server and Mongo DB databases and can be extended by re-implementing the IDatabaseService interface.

Data Synchronization

The Traceability Driver facilitates synchronization between your existing software system and the standardized data structure. The system synchronizes up to 10,000 items per batch, ensuring that your traceability data stays up to date. The synchronization process involves:

1. Automatically executing on startup.
2. Loading mappings from a local Mappings folder.
3. Transforming traceability data into standardized formats and storing it in the Data Cache.
4. After processing, the system waits one minute before the next sync cycle.

A key feature of the synchronization is that the Traceability Driver uses memory variables to track where it last left off in the data sync, ensuring that the process resumes from the correct point, rather than reprocessing the same records.

Data Cache

The Data Cache is where all traceability data is stored, making it the source for any data queries made through the API. By default, this cache is built using MongoDB, but it can be reconfigured to use other databases depending on your organization's needs.

Dashboard and Monitoring

The Traceability Driver comes with an integrated dashboard that provides visibility into the system's current state, including synchronization progress, event statistics, and errors. It allows users to:

- View synchronization progress and statistics.
- Monitor the number of events, master data entries, and syncs.
- See recent errors that have occurred during synchronization.

The dashboard also includes basic authentication, where the default password can be configured through the appsettings.json file or via environment variables.

Installation and Configuration

The Traceability Driver can be deployed in two primary ways:

1. **Docker Installation:**
    - The Traceability Driver comes as a Docker image that can be configured and deployed with ease.
    - You can use Docker Compose to set up the environment and specify how to connect to the necessary databases and configure the mappings.
2. **Release Installation:**
    - Alternatively, the Traceability Driver can be installed as a standalone module on either Windows or Linux servers. After installation, you can configure the system by editing the appsettings.json file, placing the necessary mappings in the Mappings folder, and starting the system to begin syncing the data.

Security and Authentication

The Traceability Driver supports three authentication modes:

1. OAuth (JWT) Authentication: Self-signed tokens for API access.
2. API Key Authentication: Secure the API using keys configured in the system.
3. No Authentication: For cases where authentication isn't required.

Each method provides flexibility for securing access to the system, depending on the organization's security policies.

Mapping and Event Transformation

The Traceability Driver uses mapping files to transform data from your system into a Common Event Model that conforms to the data standards. These mappings define how to query and transform data into the necessary events, master data, and traceability information.

1. Selectors are used to identify the data from your database.
2. Event Mappings define how each piece of data should be transformed into the standardized fields, such as EventId, EventTime, and ProductId.
3. Memory Variables store data between synchronization cycles to track where the last sync left off and ensure continuity.

Customizable to Fit Your Needs

The Traceability Driver offers customization to meet the unique needs of your system:

- **Multiple Database Support:** The system is designed to integrate with different databases, and custom configurations allow it to scale across various environments.
- **Dictionaries for Data Transformation:** Using dictionary-based mappings, the Traceability Driver can transform values from your database to their data standard equivalents (e.g., transforming specific equipment types or catch areas into standardized URNs).
- **Identifier Generation:** The system automatically generates traceability identifiers (e.g., EPC, GTIN, PGLN) using customizable domain settings, ensuring each piece of data is uniquely identifiable.

GDST Capability Test Integration

The Traceability Driver also integrates with the GDST's Capability Test, allowing users to verify that their implementation complies with the GDST standards. This functionality can be triggered directly from the dashboard after configuring the necessary details, such as

API keys and PGLN identifiers. You will need to request access to the GDST Capability Test by reaching out to info@theGDST.org

The Traceability Driver provides an easy, scalable, and secure solution for automating the process of becoming GDST Capable. By mapping data from existing systems into GDST events and master data, storing it in a dedicated Data Cache, and offering a fully functional GDST API, the Traceability Driver allows organizations to quickly and efficiently integrate traceability capabilities without the need for significant system modifications. Whether using Docker or traditional server installations, the system is flexible, easy to deploy, and customizable to meet your organization's specific needs.

Traceability Driver FAQs

1. What is the Traceability Driver?

The Traceability Driver is a tool that helps organizations quickly and easily comply with data standards by automatically converting and storing traceability data from your existing systems into a compliant format.

2. Do I need to be a technical expert to use it?

No, the Traceability Driver is designed to be easy to use and deploy. While it does have some configuration options, it is built to minimize complexity. Once set up, it runs automatically and requires minimal technical expertise to maintain.

3. How does the Traceability Driver help my organization?

It simplifies the process of ensuring your data is compatible with data standards. By automatically transforming and storing traceability data, it saves you time and resources compared to manually managing these requirements.

4. Where is my data stored?

Your data is stored in a separate Data Cache, which is a secure database that you control. The Traceability Driver only reads data from your existing system; it does not alter or write data back to your original system, ensuring your data remains under your control.

5. How secure is the Traceability Driver?

The Traceability Driver ensures that your data remains secure by allowing you to control access to the system via authentication methods like OAuth or API keys. Additionally, it is deployed onsite, so your data stays within your organization's network.

6. Do I need to change my existing systems to use the Traceability Driver?

No, you don’t need to overhaul your existing systems. The Traceability Driver integrates with your current systems and automatically converts the traceability data into a compliant format without requiring changes to your core infrastructure.

7. Can I choose the database for the Traceability Driver?

Yes, the Traceability Driver is flexible and can be configured to use various databases, including MongoDB, SQL Server, and MySQL. You can choose the one that works best for your organization.

8. How often does the Traceability Driver sync data?

The Traceability Driver automatically synchronizes data in batches of 10,000 items at a time, ensuring that your traceability data is up-to-date and available for querying. You can monitor this process through a simple dashboard.

9. Do I need to have any specialized hardware or software to run the Traceability Driver?

The Traceability Driver can be deployed using Docker, which allows it to run on most standard servers. No specialized hardware is required, and the system is compatible with both Windows and Linux servers.

10. Can I control who accesses the Traceability Driver?

Yes, the Traceability Driver allows you to control access to the system using OAuth or API keys. This ensures that only authorized personnel can access or manage the traceability data.

11. What happens if something goes wrong during synchronization?

The Traceability Driver includes a dashboard that shows any errors during synchronization. You can easily track issues, and the system will automatically attempt to sync again after a minute.

13. How do I set up the Traceability Driver?

You can install the Traceability Driver either via a Docker container or as a standalone release on a server. The installation process is straightforward, and detailed documentation is provided to help guide you through the setup.

14. Will I need to keep the Traceability Driver updated?

The Traceability Driver is built to be low-maintenance. However, it's a good practice to check for updates periodically. The system is designed to minimize the need for constant updates, and any necessary updates will be clearly communicated.

15. Can the Traceability Driver work with other traceability standards besides GDST**?**

The Traceability Driver was initially built for use in the seafood industry to support GDST compliance but is being extended to support other standards. Additional configurations

might be required, however the system is flexible enough for integration with various traceability frameworks.

10. What is the GDST Capability Test, and can the Traceability Driver run it?

The GDST Capability Test checks if your traceability system is compliant with GDST standards. Yes, the Traceability Driver includes the ability to execute the GDST Capability Test directly from its dashboard, making it easy for you to verify compliance.

Traceability Driver – Technology Requirements Checklist

Traceability Driver Setup Checklist

1. Existing Data Management System

☐ **Ensure you have an existing traceability or data management system** (e.g., SQL Server, MySQL, or MongoDB) where traceability data is currently stored.

☐ **Confirm the ability to connect** to this system for reading data (no write-back required).

2. Docker Setup (for Deployment)

☐ **Install Docker** on your server or machine that will host the Traceability Driver.

☐ **Ensure Docker Compose is available** for local development, if needed.

☐ **Pull the Traceability Driver Docker image** from DockerHub.

☐ **Configure your Docker container** to work with your existing systems (e.g., map the Mappings folder, set up database access).

3. Web Server Configuration

☐ **Set up a web server** (Windows or Linux server) to host the Traceability Driver API.

☐ **Deploy Traceability Driver in Docker** or as a standalone application on the server.

4. Networking Infrastructure

☐ **Ensure network access** between the existing database and the Traceability Driver.

☐ **Confirm network access** to the Data Cache for storing traceability data.

☐ **Configure firewall or proxy settings** if necessary for secure communication.

5. Security Configuration

☐ **Install SSL/TLS certificates** if you are not deploying behind a reverse proxy for secure HTTPS communication.

☐ **Set up authentication methods** (OAuth or API key) to control access to the API.

☐ **Configure secure connections** for both the Traceability Driver and the Data Cache.

6. Configuration Management

☐ **Edit the appsettings.json file** to configure the following:

- Database connections (e.g., MongoDB, SQL Server)
- API key management (if using API key authentication)
- Standardized event and master data mapping configurations

☐ **Configure environment variables** for any additional settings (e.g., URL, database settings).

7. Data Mapping Setup

☐ **Create mapping files** that define how data from your system will be transformed into compliant formats.

- Mappings should specify database selectors, event mapping fields, and field transformations.

☐ **Store mapping files** in the **Mappings folder** of the Traceability Driver installation.

8. Cloud Infrastructure Setup (Optional)

☐ **Set up cloud infrastructure** if using cloud databases (e.g., AWS, Google Cloud, or Azure).

☐ **Ensure proper cloud permissions and access** for connecting the Traceability Driver to cloud-hosted databases.

9. Dashboard Access Configuration

☐ **Access the Traceability Driver Dashboard** via a modern web browser.

☐ **Set up login credentials** for the dashboard (the default password can be configured in appsettings.json).

10. Monitoring & Error Logging

☐ **Monitor synchronization status** from the dashboard to ensure data is syncing properly.

☐ **Review error logs** for any synchronization issues and address them as needed.

11. GDST Capability Test Configuration (Optional)

☐ **Set up GDST Capability Test credentials** in the appsettings.json file if you want to run the test.

☐ **Configure the Capability Test section** with:

- GDST Test URL
- API Key
- Solution Name
- PGLN (Party Global Location Number)

12. Testing and Validation

☐ **Run the GDST Capability Test** (if configured) to verify the Traceability Driver is compliant with GDST standards.

☐ **Perform sample syncs** and verify that data is correctly transformed and stored in the GDST Data Cache.

☐ **Verify access control** to ensure only authorized users can access traceability data.

Final Steps:

- **Go live** with the Traceability Driver once the system is configured and validated.
- **Ensure regular updates** and monitor performance as data synchronization continues.