Technical Whitepaper

for the favor.ID printing application

as of 28.07.2025 for the application version 1.4.0

Content

1 Challenge

Favor.ID was created to tackle the automatic printing of ID Badges, without the need of a running GUI application and user sessions. Customers often want a designated workstation which handles the printing of badges after some criteria. These criteria are primary in form of database tables or views, which holds the data to print and some additional information such as ‘printer to use’, ‘layout to use’, print timestamp or a flag for filtering rows to print.

The printing has to be autonomic and is triggered via external software systems which have no own printing solutions.

2 Software Architecture

The software is completely written in C#, uses the .NET Framework 8.0 and runs only on a Windows OS. The reason is a third party dependency which is used to created print jobs and therefore uses Win32 functions to render and issue the print jobs.

The software consists of three parts.

- A **GUI** Application which has all functions and can edit the configuration and layouts.
- A **CLI** application, which is solely for triggering prints via a command line interface.
- The last but most important part is a **Windows Service**. This service is used to monitor a data source and automatically triggers print jobs depending on certain conditions.

2.1. Dependencies

Following dependencies are used. All dependencies are managed through nuget.net.

Package	Version
CommandLineParser	2.9.1
Microsoft.EntityFrameworkCore.Abstractions	9.0.3
Serilog.Enrichers.Environment	3.0.1
Serilog.Sinks.Console	6.0.0
Serilog.Sinks.File	6.0.0
Serilog.Sinks.MSSqlServer	7.0.2
FastReport.CedxPlugin	1.0.3
FastReport.Net	2025.1.1
libCardPrinter	2.6.0
FastReport.Localization	2025.1.1
Microsoft.Extensions.Hosting	8.0.1
Microsoft.Extensions.Hosting.WindowsServices	8.0.0
Serilog.Extensions.Hosting	8.0.0
Microsoft.SqlServer.Management.SqlParser	172.0.1
Microsoft.EntityFrameworkCore	9.0.3
Microsoft.EntityFrameworkCore.Design	9.0.3
Microsoft.EntityFrameworkCore.Sqlite.Core	9.0.3
Microsoft.EntityFrameworkCore.SqlServer	9.0.3

As mentioned above, the most important dependency is “FastReport” which supplies the designer and renders the print jobs.

2.2. Update Policy

Updates are managed offline and must be installed manually. The setup is provided as an “Inno Setup” executable, and no MSI files are supplied. The software includes a database version that is checked each time the application starts. Only the GUI application has the capability to update the database version. Other components of the software only verify the version and will not run if there is a mismatch.

3 Data Flow

The typical data flow begins with the System Database, where the main application configuration and runtime data are stored. This database securely holds all connection strings to external User Databases in an encrypted format, referred to as “Data Connections.” Additionally, any print layouts are saved within this database. The connection information for the System Database is stored encrypted in an application configuration file.

There exists a collection of User Databases that contains the data intended for printing. Each Data Connection provides information about the tables or views used as data sources. For every data source, details about each data column are stored, including its name, type, function, and length.

It is important to note that no user data is stored in the System Database, except for primary keys, table names, and column information, which are logged in the “PrintLog” table.

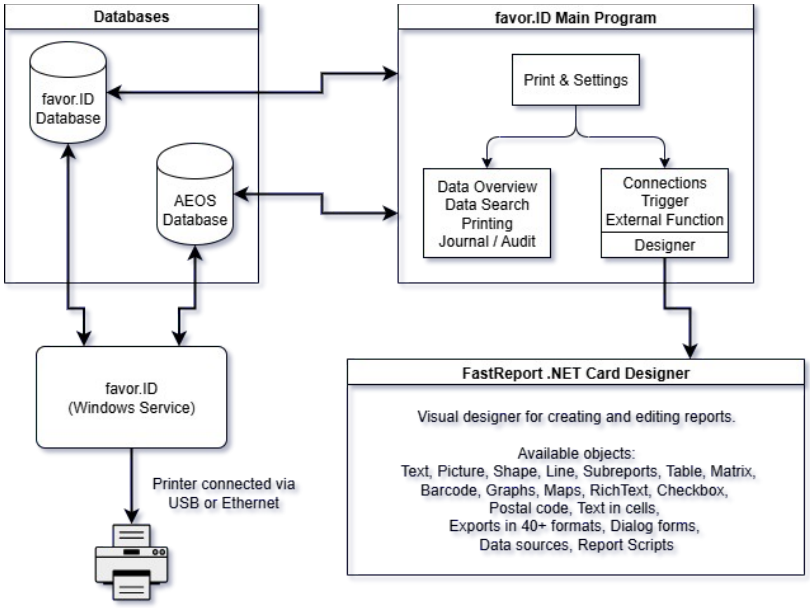
When a print job is initiated, the System Database supplies the necessary components: the Data Connection, Data Source, required column definitions, and the print layout. To prepare the print layout, one data row is retrieved from the User Database, containing only the essential columns.

Before and after the print job is prepared and sent to the printer, an entry is created in the “PrintLog” table. Each entry includes the printer name, data source name, primary key value, timestamp, error level, and a log message.

Finally, after the print job is successfully sent to the printer, data is written back to the User Database if the Data Source is configured to allow this. This data consists of the print timestamp.

3.1. Data Flow Diagram

The following Diagram shows the abstract data flow of the typical application usage.



4 Communication

The application communicates with other systems, via

- **TCP**, **Named Pipes** in case of Microsoft SqlServers.
- **PCSC** in case of CCID Encoders.
- Windows **Print Spooler** subsystem for printing.
- Printer **SDK**s for encoding commands. (Only move card to encode position commands)
- No inter process communication is used.

4.1. SqlServer

Each application part (GUI, CLI, Service) establish an own connection to both the system database and any user database. No inter process communication is used.

Communication with any database occurs either via **TCP**, using the default port **1433** for Microsoft SQL Servers or "**Named Pipes**". This communication is secured through the default SQL Server connection encryption methods. The primary method is **SSL/TLS encryption**, which encrypts the entire communication channel, ensuring that data remains confidential and protected from eavesdropping. However, the specific encryption settings and ports used depend on the SQL Server configuration chosen by the user, which may vary based on individual preferences.

All SQL commands are executed through Entity Framework Core for the System Database. In the case of the user database, commands are issued directly via SQL queries. This approach is necessary because the structure of the user data is not known at compile time, requiring queries to be generated dynamically. Each generated SQL query is checked and sanitized using "*Microsoft.SqlServer.Management.SqlParser*". This prevents illegal instructions like *DROP* or *DELETE* in the queries.

4.2. PCSC

In case of RFID the communication to the encoder is done vie the **PCSC** interface. The corresponding windows services has to be active and running.

4.3. Printers

Printer communication is managed in case of printing through the Windows Print Spooler. For RFID encoding, the printer must be accessed directly via USB, utilizing the corresponding printer SDK to position the RFID card for encoding.

The following SDKs are supported and can be utilized: Magicard SDK, Evolis SDK, Dai Nippon SDK, XID SDK, JVC SDK, Seaory SDK, and IDP Smart S70 SDK. No SDK is included with the software, so any required SDK must be installed on the target system.

5 Authentication & Authorization

Several Authentication methods are used to ensure a safe data flow or application usage.

5.1. Microsoft SqlServer Authentication

All SqlServer connections can either be authenticated via basic user and password authentication or using the Windows NT authentication of the running user. In case of the windows service, the default system service user runs the process and cant access any SqlServer with NT authentication. The user of the service must be changed in this case to a user with access rights to the database.

The main system database user should only have the “**db_owner**” and “**public**” role for the favor.ID system database.

5.2. Application Authentication

The GUI application can be secured via username and password authentication. Each user has a role, which authorizes the user. Two roles are present, “**Admin**” and “**User**”. The “Admin” role can do anything in the software, but the “User” role can only see the print log, print data and manually issue print jobs. The “User” role cannot access any settings.

The application authentication is never used in the CLI or Windows Service.

If no user is defined, which is the default, the authentication is disabled and gets activated when at least one user is created.