DARIC

710 Lakeway Drive, Suite 100
Sunnyvale, CA 94085

T (650) 260-8638
E support@daric.com

http://www.daric.com/

Configuration Guide

Version 5.0a

Daric Integration Specification

Authors:
Wes Henderson

Introduction

This document is designed to be a reference for any person wishing to implement or any person interested in integration of the Daric client application, Daric server application, dialog database, or the Daric database into an existing infrastructure. This document describes each application's architecture and sub-architecture their associated interfaces, database schemas, and the motivations behind the chosen design. Both high-level and low-level designs are included in this document.

This document should be read by an individual with a technical background and has experience reading data flow diagrams (DFDs), control flow diagrams (CFDs), interface designs, and development experience in object oriented programming and event driven programming.

This design document has an accompanying specification document and test document. This design document is per Daric System Specification version 3.0. Any previous or later revisions of the specifications require a different revision of this design document.

This document includes but is not limited to the following information for the Daric System; system overview, design considerations, architectural strategies, system architecture, policies and tactics, and detailed system design.

System Overview

The Daric application is a set of modules that encapsulate functionality for a lender application, automation, and originations platform. The application can be configured in multiple ways to suit specific clients' loan products. Daric uses the Django Web Framework (1.5) and the Python programming language (2.7), separating the functionality out into seven modules: Lender Management, Portfolio Builder, Income and Employment, Personal Credit Profiles and Applications, Accounts, Analytics, and Administrative Back-Office. Each module is written as a set of Views and helper applications corresponding to a particular set of url endpoints for that module. The front-end is templated using the Django Template Syntax, allowing for configurable web (client) applications. The server application is split between a master Web application server and a Daric Database that may be configured within the customer's MySQL installation.

The system operates under a fully permissive license, and Gazzang's zNcrypt (PCI-DSS compliant) software is used for database encryption. The log4j framework is used for intelligent system and audit logging.

Design Considerations

This section describes many of the issues that needed to be addressed or resolved before attempting to devise a complete design solution.

Assumptions and Dependencies

This design of the Daric system makes several assumptions about software and hardware, and has several software dependencies. All environmental requirements of both the database and web applications can be found in the Daric System Requirements 5.1.

Both the web server and database applications make the following assumptions about their environmental environments;

- The system can be described by the environmental requirements associated to this document.

- The system the application is executing on will have the required resources available as necessary. This entails sufficient memory and permanent storage space, an adequate CPU for the necessary application, and a TCP/IP network connection.

The web application makes the following assumptions about its operation environment;

- The client browser will have JavaScript enabled for full feature interactivity.
- The client browser will support HTML 5 for full feature interactivity
- The client is a browser on a mobile or tablet device, or computer.

The server application makes the following assumptions about its operation environment;

- The server machine will have the necessary software installed, and will be open to the Internet on ports 22, 80, and to port 3306 for appropriate private IP addresses within the VPC. These components are required for our implementation of access to the Daric database.
- The database machine will have the necessary databases setup and accessible through port 3306 from the web application.

Goals and Guidelines

The major goal of the Daric web application is that it be extremely simple and intuitive to use. The application is geared towards the prospective borrower and lender, not a technically inclined individual. It is very important that the prompts for the user be clear and concise since this will be the highest level of interaction between the application and the user. It is also important that series of prompts and responses be tested with users before being deployed as part of the product so that all interaction is "approved" by a potential user.

The second major goal of the application is that the borrower-user gets a response in a timely fashion. Intuition tells that a user will lose interest if they have to wait long times for software to respond. This is why the design has minimal data transferred between client and server or between database and application. In this design, a minimum set of information is transferred to the server in order to retrieve the necessary information, and the server only returns the requested data that is then formatted into a readable phrase on the client side.

A third major goal is that the web application be fully mobile compliant and that a mobile application for use on major mobile platforms be made available.

This design attempted to keep the web application as data independent as possible. All prompts and responses are completely data driven, so any prompt or response can be changed by a simple database change without changing any code. This makes the web application capable of prompting and responding to various structural types of data.

The Daric server is intended to have a simple interface that is relatively easy to administer. A minimal yet complete set of options is provided for the server administrator to have control of resources consumed by the server application. These options include, but are not limited to; controlling the limit of clients able to connect to the server for maximum efficiency, ability to configure which port the server listens on, ability to change the Daric database location, and control how often the database is updated.

Architectural Strategies

The Daric system design has been divided into three major sub-systems; web application, back-office application, and Daric database. The web application is then separated into six major sub-sections corresponding to the modules (see System Overview).

The web application's major design considerations include easy Daric data retrieval, easy database updates, mobile and browser client support, and a minimal set of administrative features. The server application has been designed to be as flexible as possible, trying not to design the server for specifically Daric's loan products, but for more complicated products with different servicing strategis. Given the project's constraints of human resources, software resources, and time, the server is not completely "data independent". Portions of the server application are specific to this Daric system. These portions are discussed in the server application's detailed design strategies.

Given the system's requirement that the client must be supported on Internet Explorer, Mozilla, Safari, Chrome, as well as mobile browsers supported on the Android and iOS platforms, this design attempts to balance performance considerations across all potential use cases.

System Architecture

Submodule Architecture

1 – Administrative Back Office

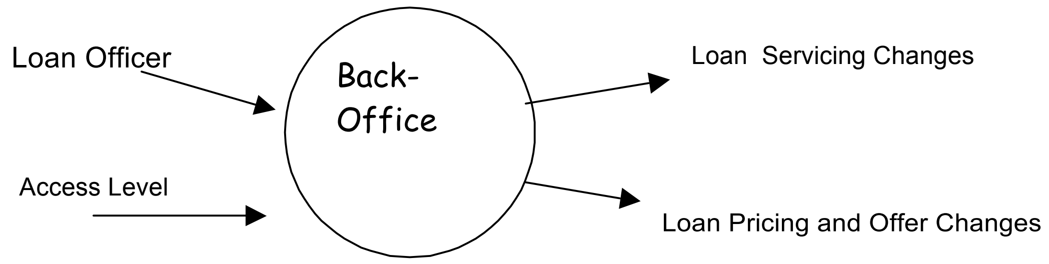


Figure 1 – Administrative Back Office

Continued breakdown

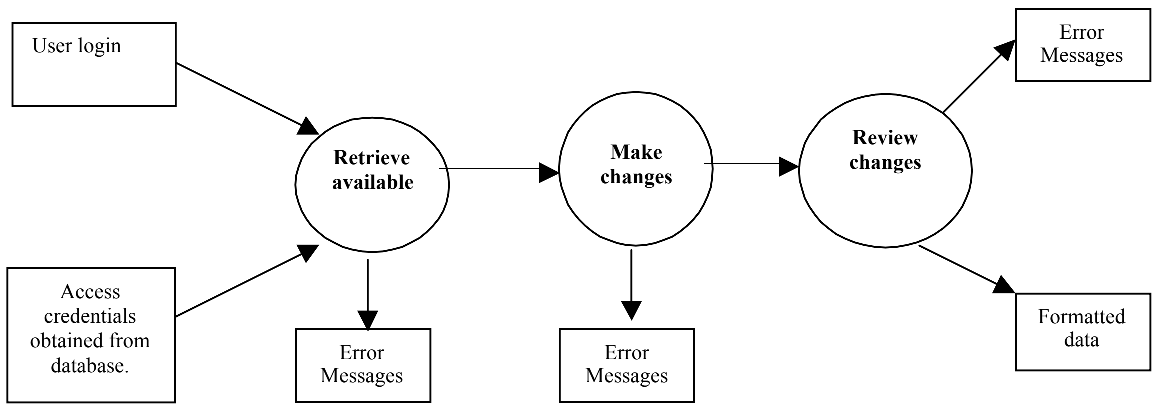


Figure 2 – Administrative Back Office Level 0

User Login:

We authenticate the user against the list of access credentials provided for back-officers and loan administrators. Once that is done, the appropriate data from the loan and account database is viewable and manipulable through Grappelli (Interface).

Retrieve Viewable Portfolio

All data elements and editable data are provided for appropriate pricing on loans, associated servicing files, Accounts with transaction level data, and Lender Portfolios for changes to the files.

Format outputs:

Puts parsed data in the format discussed in the interface section. Then check if we get the correct data. The reason that we wait until this part to check the data instead of doing that right after we get the data is efficiency. We don't want spending too much time checking data. If the data is correct, then write it to file. Otherwise, log errors.

File maintenance

Create the directories data and logerr under the directory contains the programs to store results and log errors, respectively.

There is a set of back-office CRON routines that are run nightly for housekeeping (automated servicing routines, email notifications, and payment requests, as well as updates to servicing files to reflect lastest data). Additionally the routines handle the automated compliance processes, including the dispersal of notices of action taken.

2 - Accounts

The Accounts class has two distinct Django models:

1) Accounts store bank account and payment profile information for ACH transactions.
2) Transaction objects store information about loan investments, disbursal, repayment, and servicing of a specific account.

API endpoints are provided in the Detailed System Design.

3 - Analytics

The Analytics module can be broken up into three distinct sub-components; The charts, Metrics, and the reporting framework which are all written in Python and accessible through a GUI. API endpoints are provided in the Detailed System Design

Each module contains a logger which will log information about the server application, client, or Daric database. This log is written out to file in the event that the server unexpectedly terminates. Optionally the log can be displayed in any Object. The logger has two states it operates under; debug or not debug. In debug mode, the logger will log any request to log information that is called upon it. In non-debug mode, the logger discriminates between mandatory logs, and debug logs and records only the mandatory information.

The server properties sub-component is used to store the properties and state of the server that must be maintained when the server is terminated. Such properties include; server port, maximum number of clients, debug mode, data server, and Daric data fetch time. All server properties are retrieved and stored in a properties file called “options.txt”. When an property is changed, it is written out to the properties file. In the event that a property description is not found in this file, a default is assigned to a given property.

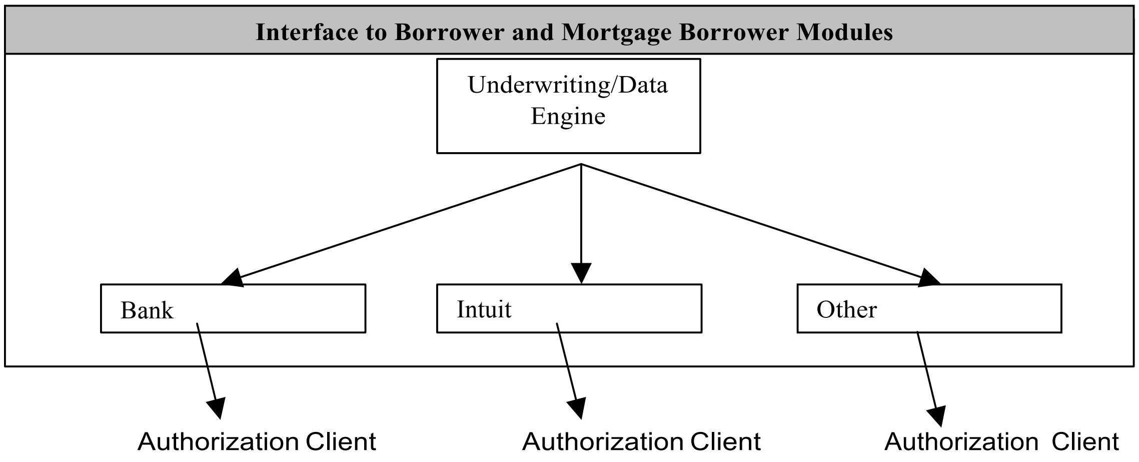
4 – Lender and Portfolio Builder

This module provides all logic routines and configurable servicing files and reports for booked loans associated with a given portfolio or lender. Data for the back office is generated here, with transaction level information on loan payments, support for reporting on portfolio performance, and calculation of metrics for portfolio building. Additionally, the Servicing sub-module which runs as an automated CRON routine (nightly at 2 am or whenever necessary) uses Loan model data from this module for updates.

Models: Loan, Note (pre-booking), Lender (portfolio summaries)

5 – Borrower Profiles Automated Verification

Internal Structure



The Mortgage Borrower and Borrower modules contain the logic and data models necessary to access, edit, create, and remove Credit and Income Profiles as well as to accept and track applications and Loans. API Endpoints are discussed below.

The Underwriting and Decisioning Engine is responsible for managing the user's authorization connections to authorize third party access to online banking, Intuit, ADP, TurboTax, and other financial data providers. When an authorization client requests a connection, the appropriate credentials for either a new account or an account refresh must be collected with the request and passed to the data service for authentication. Each client connected to the server is associated with one data account. The Underwriting and Decisioning Engine also accepts direct written authorization for credit pulls and interfaces with Experian, D&B and others for such services.

Models: CreditProfile, MortgageCreditProfile, Income Profile, Application

Policies and Tactics

This design was attempted to be made as modular as possible. This provides flexibility between component developments. In design, we attempted to partition the development into sections that each individual could create independent of another, and have a clearly defined interface between components. This would make compilation of the administrative and web applications trivial. For example, the communications components between modules work together, and are nearly independent of the data that they are transferring. With a clearly defined interface for the communications components, integration of these components is made simple and painless.

This design also took the policy of using coding standards such as standard Python variable prefixes and caption. Generally method/property purposes are easily deciphered by their descriptive name.

Detailed API Design

Daric has implemented a RESTful API that allows for querying and f iltering of loans, back-office requests and creation of accounts and borrower profiles,
among other items.

Authenticating to the API is a very simple process. Simply use basic HTTP Authentication with your Daric username and password.
Example
```
curl https:// api. daric. com/ lender/ noteset/ - u username: password
```

---

1 – Administrative Back Office

Classification

Modular administrative back office for loan products.

Purpose

This class implements the interface with the credentialing and data servers necessary to support a user-friendly back office interface. This class is used as part of the server application.

2 - Accounts

Classification

Modular accounts management on web server.

Purpose

This subsystem is designed to provide an interface to update and enter private account information and transactions associated with a particular account.

Endpoint

/account/getAccountInformation

Purpose

Method to provide a client the requested information from the server in a readable fashion.

3 – Borrower and Mortgage Borrowers

Classification

Module responsible for applications, credit and income profiles .

Purpose

This class implements any borrower interactions necessary for the underwriting process

Endpoint

/borrower/createLoanApplication(boolean bDebug, Dictionary(can be a request.POST))

Purpose – Constructor for the LoanApplication object.
return value – None

Endpoint

/borrower/getCreditProfile(boolean bDebug, borrower_id)

Purpose – Return credit and income profile for a borrower
return value – Dictionary of attributes

Endpoint

/borrower/getMortgageCreditProfile(boolean bDebug, Mortgage_borrower_id)

Purpose – Return credit and financial profile for a Mortgage Borrower
return value – Dictionary of attributes

4 – Lender

Classification

Modular servicing and lender interaction

Purpose

This class implements the lender and servicing modules.

Endpoint

contractSet(lender_id), RETURNS

ListingIdentifier	Unique key to identify the listing / loan that we bid on.
LoanIdentifier / ContractIdentifier	Unique key to identify our part of the loan we bid on whether or not it's a pending or successfully syndicated loan. We could potentially have multiple of these per ListingIdentifier where multiple bids are allowed on a partial loan.
ListingDateTimeUTC	
AccountIdentifier	Our account number at the Platform
BidDateTimeUTC	If we're able to make multiple bids on a partial loan, this will have a different time associated to each bid (as well as a different LoanIdentifier / ContractIdentifier)
LoanStartDate	
MaturityDate	
Currency	
LoanTerm	In months
Grading	A, A+, C- etc.
InterestRate	This is the annualized interest rate that we're receiving
InitialLoanAmount	
TotalPrincipalRepaid	
TotalInterestPaid	
TotalPlatformFeesPaid	The platform fee associated to the contract
TotalPenaltyFeesPaid	Any fees assessed by the platform for late payments or overpayment that's passed through to the lender
Status	Status of the loan (Pending, Cancelled, Current, Late, Defaulted, Matured)
ExpectedPayments	The total number of expected payments over the life of the loan
TotalPaymentsMade	Total payments made of the life of the loan
LatePaymentsMade	Total of payments made that were past due over the life of the loan
TotalPaymentsPastDue	How many payments behind is this loan/contract
DaysPastDue	How many calendar days past due are the payments
ExpectedAnnualLoss	Expected default rate for this credit grade on your platform (can change over time)
PlatformBorrowPower	Maximum amount you would've lent to the borrower on the platform (had they requested)
PaymentFrequency	The payment frequency for the loan
BorrowerIdentifier	The unique id associated to the borrower
Total AmountLent	The total amount lent to the borrower across all lenders (relevant for partial loans)

cashtransset(lender_id) RETURNS

LoanIdentifier / ContractIdentifier	Unique key to identify our part of the loan we bid on whether or not it's a pending or successfully syndicated loan. We could potentially have multiple of these per ListingIdentifier where multiple bids are allowed on a partial loan.
TransactionIdentifier	The platform's reference to the transaction
TransactionDate	
TransactionCurrency	
TransactionDescription	User friendly description of the transaction
Principal	Portion of payment towards principal
Interest	Portion of payment towards interest
PlatformFee	Fees charged by the platform
PenaltyFee	Fees passed through by the platform for late payment or overpayment (if any)
SettlementDate	The date you should expect the cash movement to hit your account (relevant for transfers to accounts off-platform). Otherwise, likely same as TransactionDate

lenderSet (lender_id) RETURNS

AccountIdentifier	Our account number at the Platform. As we may have several accounts
Currency	
BalanceAmount	Total cash balance in your account
TotalAvailable	Total cash available for bidding on new loans/contracts
TotalInvested	Total invested (for all active loans/contracts)
TotalPending	The cash you're expecting to move out of your account shortly as you've committed it to a loan/contract
TotalAccruedInterest	Total Accrued Interest due to you for active investments
TotalPenaltyFeesReceived	Total payments passed through to you for late payment or overpayments by the borrower (if any) (for all loans/contracts)
TotalPlatformFeesPaid	Total fees paid to the platform (for all loans/contracts)
TotalInterestReceived	Total interest payments received on your investments (for all loans/contracts)
TotalPrincipalRepaid	Total principal repaid across all your investments (for all loans/contracts)
TotalPrincipalInvested	Total of all the loans you've made (for all loans/contracts)
TotalPrincipalDefaulted	Total principal lost due to defaulted loans
InterestLostFromDefaults	Total of future interest payments lost due to defaulted loans

/noteset/?filters (request.GET)
Use Case: Get all loans that satisfy a set of credit criteria.

Purpose: The set of filters is specified as a set of query parameters. For example, https://www.daric.com/lender/noteset/?incomemin=10000 requests notes where the associated borrower has an income of $10,000 or more.

The response includes note and borrower information, including the set of 150 premier attributes as defined through Experian.

A sample response is provided below in JSON form, representing a single note:

```
{
        "id": 32,
        "borrower": {
            "premierattributes":    {
                "id": 151,
                "ALJ0316": 98,
                "ALJ5030": 999999998,
                "ALJ5320": 999999998,
                …

                "RTR2388": 1,
                "RTR3422": 0,
                "RTR6200": 0,
                "RTR7110": 1,
                "STU0300": 33
            },
            "overBorrowingLimit": true,
            "creditScore": "703",
            "grossIncome":  288000,
            "employmentLength": 3,
            "revolvingCreditUtil": 0.02,
            "earliestCredit": "1994-01-10",
            "openCreditLines":  11,
            "totalCreditLines": 15,
            "currentDelinquencies":  1,
            "currentDelinquentAmount":   0,
            "publicRecords":  0,
            "pastTwoYearsDelinquencies": 1,
            "homeownerStatus": " ",
            "pastSixMonthsInquiries": 0
        },
        "grade": "B4",
        "interestRate": 0.13,
        "interestRatePercent": "13.00%",
        "amount": 8000,
        "term": 36,
        "creditScore": "703",
        "title": "Debt Loan",
        "description": "",
        "purpose": "debt consolidation",
        "amountRemaining": "8000.00",
        "percentFunded": "0%",
        "expiryDate": "2014-01-06"
    }
```

The filter set specified in the URL may set values for incomeMin, incomeMax, creditScoreMin, creditScoreMax, and maximum threshold values for any attribute in the Premier Attribute Set. Simply chain together parameters. The full data is provided for customized score calculation purposes.

For example, https://www.daric.com/lender/noteset/?incomemin=10000&ALJ0316=1 yields notes that satisfy the income criterion as well as the requirement that the attribute ALJ0316 be below 1.