Technical Product Requirements Document: TrueLayer Credit Card Integration Architecture for Resolve 2.0

1. Executive Summary and Strategic Alignment

1.1 Architectural Imperative

The financial technology landscape is currently navigating a critical transition from deterministic, rule-based categorization to probabilistic, context-aware intelligence. The Resolve 2.0 platform represents a pioneering effort to bridge this divide through a "Two-Brain Architecture".[1] This Neuro-Symbolic approach delegates rigid mathematical optimization—specifically debt payoff schedules and interest minimization—to a deterministic "Math Brain" powered by Google OR-Tools, while assigning transaction enrichment, categorization, and anomaly detection to a probabilistic "Agentic Brain" driven by Large Language Models.[2]

To function effectively, this dual-system architecture requires a continuous, high-fidelity stream of financial data. For the "Math Brain," this means precise integer balances, interest rates, and minimum payment definitions.[1] For the "Agentic Brain," it necessitates rich transaction metadata including merchant names, timestamps, and amounts.[2] TrueLayer has been selected as the critical infrastructure partner to bridge the user's financial reality with these internal systems.

This Product Requirements Document (PRD) serves as the definitive technical specification for integrating TrueLayer's Data API into the Resolve platform. The primary objective is to enable the secure, reliable, and compliant extraction of **credit card transaction and balance data**. Unlike standard Open Banking integrations that primarily target current accounts (checking), this integration focuses exclusively on **liability accounts**. This distinction necessitates a rigorous configuration of TrueLayer's Authentication Link (Auth Link), strict scope management, and precise provider filtering to prevent user error during the onboarding process.

1.2 The "Asset vs. Liability" Data Challenge

The core mission of the "Paydown Pilot" module is to empower users to minimize total interest and become debt-free faster.[1] To achieve this, the optimization engine requires a strict separation of assets (debit accounts) from liabilities (credit cards). A fundamental risk in Open Banking integrations is the "Account Pollution" scenarios, where a user inadvertently connects a debit account instead of a credit card. If the "Math Brain" ingests a positive asset balance as

a debt obligation, the optimization model will fail to converge or produce erroneous payoff schedules.

Therefore, this integration is not merely about connectivity; it is about **filtered connectivity**. The system must enforce constraints at the very entry point—the Authentication Link—to ensure that only credit card providers and data scopes are presented to the user. This requires a sophisticated utilization of TrueLayer's cards scope hierarchy, which is distinct from the standard accounts scope used for payment initiation or asset aggregation.[3]

1.3 Scope of Implementation

This document outlines the technical requirements for deploying this integration within a **Replit-hosted Python (FastAPI)** environment. It covers:

- **Authentication Flow:** Configuration of the Hosted Auth Link using the TrueLayer Console and dynamic URL construction to enforce scope restrictions.[5]
- **Scope Management:** Implementation of the specific cards scope family (cards, cards:transactions, balance) to access liability data while adhering to data minimization principles.[7]
- **Provider Filtering:** Strategies to restrict the banking selection UI to credit card issuers (e.g., Amex, Barclaycard, MBNA) using specific provider_id filters.[8]
- **Data Ingestion:** Mapping the /cards endpoint family to the internal Account and Transaction data models required by the Resolve platform.[3]
- **Replit Deployment:** Considerations for managing environment variables, secrets, and callback handling within the Replit infrastructure.[10]

---

2. Authentication Architecture and Auth Link Configuration

2.1 The Authentication Paradigm

TrueLayer operates on an OAuth2-based authentication model. For the Resolve platform, utilizing the **Full Flow** or **DIY Bank Selection** models is mandated. The "Full Flow" allows TrueLayer to handle the regulatory complexities of consent screens, while the "DIY Bank Selection" offers the granular control necessary to filter providers before the user enters the TrueLayer environment.[6]

The entry point for any user interaction is the **Auth Link**. This is not a static URL but a dynamically constructed endpoint that dictates the terms of the data exchange. For the Resolve platform, constructing this URL requires precise parameterization to target credit card data specifically.

2.2 URL Construction and Parameter Specification

The Auth Link is a parameterized URL pointing to https://auth.truelayer.com/. The construction requires strict adherence to specific parameters to enforce the "Credit Card Only" constraint. The following specification defines the required parameters for the Resolve implementation.

2.2.1 Base Configuration

The fundamental structure of the Auth Link is as follows:

HTTP

```
https://auth.truelayer.com/?
  response_type=code&
  client_id=<YOUR_CLIENT_ID>&
  redirect_uri=<YOUR_REDIRECT_URI>&
  scope=<SPACE_SEPARATED_SCOPES>&
  nonce=<UNIQUE_STRING>&
  state=<CSRF_TOKEN>&
  providers=<PROVIDER_LIST>
```

2.2.2 Critical Parameter Logic

The response_type must be set to code.[8] This mandates the OAuth2 Authorization Code Grant flow. This is a non-negotiable security requirement. It ensures that the sensitive access_token and refresh_token are never exposed to the frontend (React/Next.js) or the user's browser. Instead, the browser receives a temporary, one-time-use authorization code, which is exchanged for tokens via a secure, server-to-server call from the Replit backend (FastAPI).[11]

The client_id serves as the unique identifier for the Resolve application within the TrueLayer ecosystem. This ID is generated in the Console and must be stored as a secure environment variable in Replit (TRUELAYER_CLIENT_ID).[11]

The redirect_uri defines the callback endpoint on the Resolve backend (e.g., https://api.resolve.app/callback). This URI must be explicitly allowlisted in the TrueLayer Console. Any mismatch between the URI sent in the Auth Link and the one configured in the Console will result in an immediate redirect_uri_mismatch error, halting the flow.[11]

The state parameter is crucial for security. It must be a cryptographically random string (nonce) generated by the backend before generating the URL. This string acts as a Cross-Site Request Forgery (CSRF) token. When TrueLayer redirects the user back to the application, it

returns this state value. The backend must verify that the returned state matches the one stored in the user's session. If they do not match, the request may be a malicious attempt to inject an authorization code, and the process must be aborted.[8]

2.3 Deep Dive: The scope Parameter Strategy

The most significant configuration for this integration is the scope parameter. TrueLayer distinguishes strictly between "Accounts" (typically current/checking accounts) and "Cards" (credit cards). A failure to specify the correct scope hierarchy will result in the API returning access_denied or simply omitting the required data.

2.3.1 The "Cards" Scope Hierarchy

To extract the data required for the "Math Brain" (balances) and "Agentic Brain" (transactions), the following composite scope string must be used:

info cards cards:transactions balance offline_access

Scope Breakdown:

- **cards**: This is the foundational scope. It grants access to the /data/v1/cards endpoint. This endpoint returns high-level card metadata, including the card network (Visa/Mastercard), the partial card number (PAN, usually the last 4 digits), and the card type (Credit/Charge). Without this scope, the system cannot identify the existence of a credit card account.[7]
- **cards:transactions**: This scope extends the permission to the /data/v1/cards/{id}/transactions endpoint. It allows the extraction of transaction history, which is the raw material for the "Agentic Brain." The system relies on this data to identify spending patterns, recurring subscriptions, and merchant categories.[3]
- **balance**: (Note: In some contexts, this may be referred to as cards:balance, but the generic balance scope is often used in conjunction with cards). This scope enables access to /data/v1/cards/{id}/balance. This endpoint provides the *current balance* (the amount owed) and the *available balance* (the remaining credit limit). These two integers are the critical inputs for the "Math Brain." By subtracting the current balance from the sum of current and available balances, the system can derive the user's total credit limit and Utilization Ratio—a key metric for credit score optimization strategies.[1]
- **offline_access**: This is essential for the long-term viability of the Resolve platform. When granted, this scope instructs TrueLayer to return a refresh_token alongside the short-lived access_token. The access_token typically expires after one hour. Without a refresh token, the user would be forced to re-authenticate every time the "Math Brain" needed to re-run an optimization. The offline_access scope allows the backend to silently refresh tokens in the background for up to 90 days (in the UK), enabling continuous monitoring and "Agentic" alerts without user intervention.[7]
- **info**: This scope retrieves the user's identity data, such as their full name. While

secondary to the optimization logic, it is useful for verification purposes, ensuring the connected card belongs to the authenticated user.[7]

2.3.2 The Exclusion of accounts

It is a strategic architectural decision to **exclude** the accounts scope from the default Auth Link. The accounts scope targets current accounts, savings accounts, and other asset-based products.[12] Including it would clutter the user's selection screen with non-relevant banking products, increasing the cognitive load and the risk of connecting the wrong account type. While some providers (like Lloyds or Monzo) bundle credit and debit products under a single login, the cards scope acts as a data filter. Even if a user authenticates with a provider that holds both their mortgage and their credit card, requesting *only* the cards scope ensures that TrueLayer's API surfaces only the credit card data, keeping the "Math Brain" free of asset-side noise.[7]

2.4 Console-Side Configuration Constraints

Configuring the URL parameters is necessary but insufficient. TrueLayer enforces a "double-lock" mechanism where scopes must be enabled in two places: the application code (via the URL) and the TrueLayer Console.

The engineering team must navigate to the **Data API > Settings** or **Auth Link Builder** section of the Console. Within the **Permissions** tab, the checkboxes for **Cards**, **Transactions**, **Balance**, and **Offline Access** must be explicitly enabled. If these are not ticked in the Console, the API will reject the Auth Link request with an invalid_scope error, even if the URL is syntactically perfect.[13]

---

3. Provider Selection and Filtering Strategy

3.1 The Friction of Universal Selection

The default behavior of the TrueLayer Auth Dialog (when providers=uk-ob-all is used) is to display every supported banking entity in the UK ecosystem. For a specialized application like "Paydown Pilot," this creates significant user friction. A user searching for "Tesco" might be presented with "Tesco Bank" (Current Account) and select it, only to find their credit card data missing because the connector used did not support the cards scope. This "silent failure" erodes user trust.

To mitigate this, the integration must leverage TrueLayer's provider_id parameters to curate the experience, effectively creating a "walled garden" of supported credit card issuers.

3.2 Filtering via Provider IDs

The Resolve platform will implement a "DIY Bank Selection" flow.[6] In this model, the user

selects their credit card provider *within the Resolve app interface*, and the app then generates an Auth Link specific to that provider. This bypasses the generic TrueLayer selection screen entirely.

3.2.1 The "Credit Card Only" Filter Implementation

To support this flow dynamically, the Resolve backend must query the TrueLayer /providers endpoint to retrieve a verified list of supported institutions.

API Call for Dynamic Filtering:

```
HTTP
```

```
GET https://auth.truelayer.com/api/providers?scope=cards&country=uk
```

By passing scope=cards as a query parameter, TrueLayer filters the response to return *only* the providers that explicitly support credit card data retrieval.[14] This eliminates the risk of a user selecting a bank that only offers current account access. The response from this call should be cached (e.g., in Redis or an in-memory dictionary) to minimize latency, as provider capabilities change infrequently.

Once the user selects a provider from this filtered list (e.g., "American Express"), the frontend sends this selection to the backend. The backend then constructs the Auth Link, appending the specific provider_id:

...&provider_id=ob-amex

This parameter forces the Auth Dialog to skip the selection screen and land the user directly on the consent page for American Express, significantly reducing friction and error rates.[8]

3.3 UK Credit Card Provider Ecosystem Analysis

Successful integration requires handling the specific idiosyncrasies of each provider. The following analysis details the provider_id mappings and known quirks for major UK credit card issuers.

Provider Specifics Table [3]:

Provider Name	Provider ID (TrueLayer)	Integration Notes &

		Quirks
American Express	ob-amex (or oauth-amex)	**Complexity:** Returns supplementary card data as metadata objects. It does not always return a discrete balance for supplementary cards, as the liability sits with the primary account holder. The "Math Brain" must be logic-aware to aggregate supplementary transactions under the main account balance.[3]
Barclaycard	ob-barclaycard	**Critical Distinction:** This is distinct from ob-barclays, which is for current accounts. Connecting ob-barclays with the cards scope will likely fail or return endpoint_not_supported. The UI must explicitly label this as "Barclaycard" to distinguish it from "Barclays Bank".[15]
Capital One	ob-capital-one	**High Fidelity:** Known to support both date and time in transaction timestamps, allowing for more precise event correlation by the "Sherlock" module.[15]
MBNA	ob-mbna	**Timestamp Limitation:** Although part of Lloyds Banking Group, it has a distinct ID. MBNA often returns transactions with timestamps set to midnight

		(00:00:00), reducing the precision of the "Agentic Brain's" time-of-day analysis.[15]
Virgin Money	ob-virgin-money	**Known Issue:** Users often encounter a 414 Request URI Too Large error on desktop browsers. This occurs because the authentication flow relies on an app-to-app handoff. The Resolve UI should detect this provider and prompt the user to complete the connection via their mobile device.[18]
Tesco Bank	ob-tesco	Fully supports credit card data retrieval via Open Banking.
Sainsbury's Bank	ob-sainsburys	Similar to MBNA, often returns date-only timestamps.[15]
NewDay	ob-newday (varies)	NewDay underpins store cards for Amazon, John Lewis, and others. The integration may require selecting the specific retail brand if a generic ID isn't used. Testing is required to verify if ob-newday covers all co-branded cards.
Lloyds / Halifax	ob-lloyds, ob-halifax	**Bundled Access:** These providers bundle current accounts and credit cards. Connecting via ob-lloyds with the cards scope *will*

		return credit cards held with Lloyds. The backend must be prepared to filter the results if the user holds multiple product types.[3]

---

4. Data Extraction: The /cards Endpoint Family

4.1 The Divergence from /accounts

A common pitfall in TrueLayer integrations is attempting to use the /accounts endpoint as a catch-all. For credit cards, the /accounts endpoint will often return an empty list, an error, or strictly asset data. The Resolve platform must strictly utilize the /cards endpoint family for all liability data extraction.[3]

4.2 Endpoint Architecture and Data Models

Once the access_token is obtained via the Auth Link flow, the "Agentic Brain" initiates data extraction. The retrieval process follows a specific hierarchy.

4.2.1 Listing Cards

Request:
GET /data/v1/cards
Header: Authorization: Bearer <ACCESS_TOKEN>
Response Analysis:
The response returns an array of card objects. Key fields for the "Math Brain" include:

- account_id: The unique, persistent identifier for the card. This ID is required for all subsequent API calls (balance, transactions).
- card_network: Returns values such as VISA, MASTERCARD, or AMEX. This is critical for the frontend UI to display the correct card logo, reinforcing user trust.[3]
- card_type: Returns CREDIT or CHARGE. This field is vital for the "Math Brain" logic. A CHARGE card (often Amex) *must* be paid in full every month, representing a "hard constraint" in the optimization model. A CREDIT card allows revolving debt, which is the primary target for interest minimization strategies.[3]
- currency: (e.g., GBP). The optimization engine operates on integer cents. This field confirms the currency base to ensuring the correct decimal conversion is applied.[1]

4.2.2 Fetching Balances

Request:
GET /data/v1/cards/{account_id}/balance
Data Model Mapping:

- current: The total amount currently owed on the card. This maps directly to the current_balance variable in the Google OR-Tools optimization engine. It represents the immediate liability.
- available: The remaining credit limit on the card.
- **Derived Insight:** By adding current + available, the system can approximate the total credit_limit. From this, the **Credit Utilization Ratio** is derived (current / credit_limit). If this ratio exceeds 30%, the "Agentic Brain" can trigger a specific advisory warning, or the "Math Brain" can switch to a "Credit Score Optimization" strategy that prioritizes paying down high-utilization cards first, regardless of APR.[1]

4.2.3 Fetching Transactions

Request:
GET /data/v1/cards/{account_id}/transactions?from=YYYY-MM-DD&to=YYYY-MM-DD
Transaction Categorization Context:
The response includes description, amount, and transaction_category.3

- **The "Agentic Brain" Role:** TrueLayer's transaction_category provides a high-level classification (e.g., "Shopping"). However, the Resolve 2.0 Agentic Brain relies on the raw description (e.g., "UBER *TRIP", "NETFLIX.COM") to perform deeper, LLM-driven enrichment. This allows the system to distinguish between "Transport" and "Event Travel," or "Subscription" and "One-off Purchase".[19]
- **Timestamp Nuance:** As noted in the provider analysis, banks like Lloyds and Santander often return timestamps as 00:00:00 (midnight). The "Sherlock" module (event correlation) must account for this lack of temporal precision. When correlating a transaction with a real-world event (e.g., a concert), strict hour-matching must be relaxed for these providers.[15]

---

5. Technical Implementation Specification for Replit

5.1 Architecture Overview

The Resolve platform operates on a Replit-hosted architecture utilizing **Python (FastAPI)** for the backend and **PostgreSQL (Supabase)** for persistence. This environment requires specific handling of secrets, callbacks, and token storage to maintain the "Iron Wall" security posture.[2]

5.2 Environment Configuration (Replit Secrets)

Security in Replit relies on the **Secrets** tool (environment variables). Hardcoding credentials is strictly prohibited. The following variables must be configured:

- TRUELAYER_CLIENT_ID: The unique application ID.
- TRUELAYER_CLIENT_SECRET: The confidential secret key used for token exchange.
- TRUELAYER_REDIRECT_URI: The callback URL (e.g., https://<REPLIT_URL>/callback).
- SUPABASE_URL & SUPABASE_KEY: Database connection credentials.

- ENCRYPTION_KEY: A symmetric key (e.g., Fernet) used to encrypt access and refresh tokens before storage.

5.3 Backend Implementation Logic

The backend must handle the OAuth2 flow, scope enforcement, and data retrieval. The following Python/FastAPI specification outlines the critical logic paths.

5.3.1 Auth Link Generation Endpoint

This endpoint generates the secure URL for the frontend. It strictly enforces the cards scope.

Python

```python
from fastapi import APIRouter
import os
import secrets

router = APIRouter()

@router.get("/connect/credit-card")
def generate_auth_link(provider_id: str = None):
    base_url = "https://auth.truelayer.com/"

    # CRITICAL: Define Scopes strictly for Credit Cards
    # 'cards' for account info, 'cards:transactions' for history,
    # 'balance' for debt amount, 'offline_access' for 90-day refresh
    scopes = "info cards cards:transactions balance offline_access"

    # Generate secure state to prevent CSRF
    state_token = secrets.token_urlsafe(32)
    # Store state_token in DB/Cache linked to session for validation later

    auth_url = (
        f"{base_url}?"
        f"response_type=code&"
        f"client_id={os.environ.get('TRUELAYER_CLIENT_ID')}&"
        f"scope={scopes}&"
        f"redirect_uri={os.environ.get('TRUELAYER_REDIRECT_URI')}&"
        f"state={state_token}&"
        f"providers=uk-ob-all" # Default to curated list if no ID provided
```

```
    )

    if provider_id:
        # Filter strictly for the requested card provider to skip selection screen
        auth_url += f"&provider_id={provider_id}"

    return {"url": auth_url}
```

5.3.2 Data Sync Service

This service handles the retrieval of data once the token is exchanged. It demonstrates the branching logic required to handle potential edge cases where a provider might return mixed account types.

Python

```
import requests
from fastapi import HTTPException

def sync_user_cards(user_id: str, access_token: str):
    headers = {"Authorization": f"Bearer {access_token}"}

    # 1. Fetch Cards
    # We query the /cards endpoint specifically.
    cards_response = requests.get("https://api.truelayer.com/data/v1/cards", headers=headers)

    if cards_response.status_code == 404:
        # Handle case where user has no cards with this provider
        return {"message": "No credit cards found."}

    if cards_response.status_code!= 200:
        raise HTTPException(status_code=400, detail="Failed to fetch cards")

    cards_data = cards_response.json().get("results",)

    for card in cards_data:
        # Strict Filtering: Ensure we are processing a CREDIT or CHARGE card
        if card.get("card_type") not in:
            continue
```

```
        card_id = card["account_id"]

        # 2. Get Balance for Math Brain
        balance_res = requests.get(f"https://api.truelayer.com/data/v1/cards/{card_id}/balance",
headers=headers)
        balance_data = balance_res.json().get("results",)

        current_balance = balance_data.get("current", 0)
        available_balance = balance_data.get("available", 0)

        # 3. Get Transactions for Agentic Brain
        # Dynamic date range: last 90 days for initial sync
        tx_res = requests.get(f"https://api.truelayer.com/data/v1/cards/{card_id}/transactions",
headers=headers)
        transactions = tx_res.json().get("results",)

        # Store Data:
        # - Update 'accounts' table with current_balance
        # - Insert 'transactions' table with enriched data
```

---

6. Operational Resilience, Compliance, and Error Handling

6.1 Regulatory Compliance (UK/SCA)

As the Resolve platform operates in the UK, strict adherence to **Strong Customer Authentication (SCA)** is mandatory.

- **90-Day Re-authentication Rule:** The offline_access scope provides a Refresh Token that allows the application to access data without user intervention. However, under UK Open Banking standards, this consent is valid for a maximum of 90 days. After this period, the access token will expire, and the refresh token will no longer function until the user re-authenticates via the Auth Link.[20]
- **Proactive UI Messaging:** To prevent service interruption, the Resolve dashboard must track the consent_expires_at timestamp (available in the /me endpoint metadata or token response). The UI should display a "Reconnect Bank" warning to the user 7 days prior to expiration. If the token expires, the "Math Brain" will be calculating payoff plans based on stale balance data, rendering the output invalid.

6.2 Handling "Endpoint Not Supported" Errors

A specific error, endpoint_not_supported, occurs if the application attempts to query the /accounts endpoint for a purely credit-card-based provider (e.g., ob-barclaycard).

- **Mitigation Strategy:** The backend logic must be strictly typed. Providers identified as "Card Only" in the internal database must be flagged. For these providers, the system must *never* attempt to call /accounts. The sync logic should branch based on the provider type:
  - IF provider_type == 'CREDIT_ONLY': call /cards
  - IF provider_type == 'MIXED': call /cards AND /accounts
  - This logic prevents wasted API calls and error log noise.[17]

6.3 Managing Token Encryption

The "Iron Wall" privacy architecture requires that no sensitive banking credentials (even tokens) be stored in plaintext.

- **Implementation:** Upon receiving the access_token and refresh_token from TrueLayer, the backend must encrypt these strings using a symmetric key (e.g., via the cryptography Python library) before saving them to the Supabase database. They should only be decrypted transiently in memory when making an API call.

---

7. Conclusion

Integrating TrueLayer for credit card data within the Resolve 2.0 platform requires a fundamental shift from generic aggregation to targeted data extraction. By rigorously enforcing the cards scope, implementing robust provider filtering, and designing specific error handling for provider-specific quirks, the platform ensures the high-fidelity data ingress necessary for its advanced AI components.

This architecture ensures the "Math Brain" receives precise, isolated liability data for deterministic optimization, while the "Agentic Brain" receives the granular transaction detail needed for contextual reasoning. This fulfills the core promise of the Resolve 2.0 platform: to provide a mathematically optimal, contextually aware path to financial freedom.

8. Appendix: Verified UK Credit Card Provider Configuration

Provider	TrueLayer ID	Supported Scopes	Notes
American Express	ob-amex	cards, balance, transactions	Metadata for supplementary

			cards.
Barclaycard	ob-barclaycard	cards, balance, transactions	Distinct from ob-barclays.
Capital One	ob-capital-one	cards, balance, transactions	Accurate time support.
MBNA	ob-mbna	cards, balance, transactions	Midnight timestamps.
Tesco Bank	ob-tesco	cards, balance, transactions	
Sainsbury's Bank	ob-sainsburys	cards, balance, transactions	Midnight timestamps.
Virgin Money	ob-virgin-money	cards, balance, transactions	Mobile app handoff critical.
Lloyds Bank	ob-lloyds	cards, balance, transactions	Mixed product provider.
Halifax	ob-halifax	cards, balance, transactions	Mixed product provider.
NatWest	ob-natwest	cards, balance, transactions	Mixed product provider.
Nationwide	ob-nationwide	cards, balance, transactions	Mixed product provider.
Santander	ob-santander	cards, balance, transactions	Mixed product provider.
HSBC	ob-hsbc	cards, balance, transactions	Mixed product provider.

Works cited

1. Credit Card Optimiser Gem Training Overview
2. Refining Event Matching and PDF Parsing
3. Card data requests - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/card-data-requests
4. Account and card data - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/account-and-card-data
5. Data API in Console - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/data-api-in-console
6. Customise the auth journey - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/customise-the-auth-journey
7. Scopes - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/scopes
8. Generate an auth link - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/generate-an-auth-link
9. How can I select specific providers in my Auth Dialog? - TrueLayer, accessed on January 11, 2026, https://support.truelayer.com/hc/en-us/articles/360019754134-How-can-I-select-specific-providers-in-my-Auth-Dialog
10. Credit Card Optimiser PRP 3
11. Connect an account - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/connect-an-account
12. Account data requests - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/account-data-requests
13. Build a Data auth link - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/build-data-auth-links
14. List providers - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/reference/getproviders
15. Which providers return the hour in the timestamp? - TrueLayer, accessed on January 11, 2026, https://support.truelayer.com/hc/en-us/articles/6372056993041-Which-providers-return-the-hour-in-the-timestamp
16. [Amex] What do I need to know about Amex supplementary cards? - TrueLayer, accessed on January 11, 2026, https://support.truelayer.com/hc/en-us/articles/6370652642833--Amex-What-do-I-need-to-know-about-Amex-supplementary-cards
17. What does 'endpoint_not_supported' mean? – Help Centre - TrueLayer, accessed on January 11, 2026, https://support.truelayer.com/hc/en-us/articles/360004569718-What-does-endpoint-not-supported-mean
18. Why do I get a '414 Request – URI Too Large' error when connecting to Virgin Money?, accessed on January 11, 2026, https://support.truelayer.com/hc/en-us/articles/8066469779345-Why-do-I-get-a-

414-Request-URI-Too-Large-error-when-connecting-to-Virgin-Money
19. Technical Report: Resolve 2
20. Connections - TrueLayer documentation, accessed on January 11, 2026, https://docs.truelayer.com/docs/connections