



API INTEGRATION CONTRIBUTOR GUIDE

Inverité Microcheck

A developer guide to reporting loan requests from your systems into CreditBook Microcheck, via the Inverité Microcheck integration.

The official CreditBook Microcheck API Documentation holds the current API parameters, request and response structures, and status/reason codes. This guide supplements it; it is not API documentation.

Prepared by	Operations Team
Document owner	Inverité
Version	1.0
Classification	Public
Date	2026-09-22

CONTENTS

1.	About this guide.....	2
2.	Getting started.....	2
3.	How the integration works.....	3
4.	Sandbox and production.....	4
5.	Integration flow.....	5
6.	Reporting loan requests.....	6
7.	Developer implementation guidance.....	7
8.	Testing and sandbox guidance.....	8
9.	Official API documentation.....	9

1. About this guide

This guide explains how the Inverite Microcheck integration with CreditBook Microcheck fits together, and how your developers report loan request data into it. It covers the contributor (write) side only: creating, updating, and status-tracking your own Loan Requests.

What this guide contains

- How the Inverite-CreditBook relationship works, and where each party's responsibility starts and stops.
- The end-to-end integration flow: when to create, update, and report status on a Loan Request.
- Sandbox and production concepts, and guidance on testing before go-live.
- Practical implementation guidance and a sequencing example.

What this guide does not contain

This document is not API documentation. It does not list every request parameter, response field, schema, or status/reason code. Those are published in the official CreditBook Microcheck API Documentation and are not repeated here.

Refer to the official CreditBook Microcheck API Documentation for the current API parameters, request and response structures, and status/reason codes.

docs.creditbook.ca/api

For account setup, your API key, or anything about your Microcheck agreement, contact Inverite. For questions about CreditBook's API behavior itself, contact the CreditBook team directly.

2. Getting started

2.1 Prerequisites

- A Microcheck contributor agreement with Inverite.
- An API key. The same key works in sandbox immediately; CreditBook enables it for production once your integration is ready to go live (Section 4).
- Familiarity with the Loan Request lifecycle in Section 5 before you write any code.

2.2 Authentication

Every request to the CreditBook Microcheck API is authenticated with your API key, sent as a request header:

```
X-API-Key: YOUR-API-KEY-HERE
```

Your API key is the same value in sandbox and production. It is not reissued when you go live. CreditBook must separately enable it for production use; see Section 4 for how to request that.

Refer to the official CreditBook Microcheck API Documentation for the current authentication header format and error responses.

docs.creditbook.ca/api

2.3 Environments

There are two environments: sandbox, for integration and testing, and production, for live merchant data. Integrate and test against sandbox first. Section 4 covers the differences and what to change when you move across.

3. How the integration works

Your contract for Microcheck is with Inverite, but the day-to-day API calls your developers make go straight to CreditBook. Here's how that fits together:

Why direct to CreditBook?

CreditBook Microcheck is a shared database that many lenders contribute to at once. For that data to be attributed correctly and stay reliable in real time, each contributing lender submits under its own API key directly to CreditBook. That's simply how the platform is built, regardless of who arranged the access. Inverite's role is to get you set up with that access and support the relationship, not to sit in the middle of the data flow.

In practice: come to Inverite for onboarding, your API key, or anything about your Microcheck agreement. Go to CreditBook (or this guide and their documentation) for questions about API behavior itself.

Step	What happens
1. Onboarding	You sign up for Microcheck with Inverite. Inverite handles the commercial relationship. There's no separate contract or billing with CreditBook.
2. Access request	Inverite reaches out to CreditBook on your behalf and requests contributor access to CreditBook Microcheck.
3. Credential issuance	CreditBook provisions an API key scoped to your organization and returns it to Inverite.
4. Retrieve your key	Once the key is ready, log in to the merchant portal and navigate to API > Settings > Vendor API Keys . From there, you can view and copy the credentials for the Microcheck reporting integration.
5. Direct integration	Your developers build against CreditBook's API directly (the sandbox/production URLs in Section 4) using that key. There is no Inverite gateway or proxy in between.
6. Go live	Once your integration is complete and ready for production, email merchantsupport@inverite.com . Inverite reaches out to CreditBook to enable your API key for production use. It's the same key you already have.

4. Sandbox and production

Request and response structures are the same in both environments. What changes when you move to production is the base URL. Your API key stays the same, but CreditBook must enable it for production use first.

Environment	Base URL	Monthly requests	Rate
Sandbox	https://sandbox.api.creditbook.ca/v1	10,000	2 req/sec (40 burst)
Production	https://api.creditbook.ca/v1	50,000	2 req/sec (40 burst)

Both environments are rate-limited at the API key level (HTTP 429 once hit); monthly quotas reset on the 1st of each month.

Migration considerations

- Your API key is the same in both environments. CreditBook must enable it for production before requests against the production base URL will succeed.
- requestId values you assign are yours in both environments, but the CreditBook-generated id for a sandbox record is not valid in production.
- Run the full lifecycle in sandbox first: create, update, status changes, and renewals (Section 5).

Go live

Once your integration is complete and ready for production, email merchantsupport@inverite.com. Inverite reaches out to CreditBook to enable your API key for production use.

merchantsupport@inverite.com

5. Integration flow

The flow is the same for every Loan Request you contribute. Each event below should trigger an API call so the shared database reflects reality in near real time.

The four steps

- 1. Create the Loan Request.** Call Create when a customer submits a new application. The response contains the CreditBook-generated id. Store it against your own requestId.
- 2. Keep it updated.** Any change to contact info, balance, or comments should be pushed as an update, individually or in a batch.
- 3. Report status.** A lending decision or life-cycle event (approved, refused, sent to collections, paid in full, renewed, etc.) is reported through the dedicated status call, not the general update call.
- 4. Retrieve or audit.** Confirm what CreditBook has on file, or pull the revision history, at any time.

Event in your system	API call to make
New loan application submitted	POST /loanrequests: creates the record. Store the returned id for all later calls.
Any detail changes (contact info, balance, comments, etc.)	PUT /loanrequests/{requestId}: updates the record and creates a new revision.
Syncing many records at once (e.g. nightly batch)	PUT /loanrequests: same effect as individual updates, done asynchronously.
A lending decision or life-cycle event (approved, refused, cancelled, sent to collections, paid in full, renewed, etc.)	POST /loanrequests/{requestId}/status: records a new status with timestamp and reason.
Confirming what CreditBook has on file, or an audit trail	GET /loanrequests/{requestId} and /revisions

Renewals: set `referenceId` on the new Loan Request to the prior loan's CreditBook id. When the new loan's status is set to accepted, CreditBook auto-updates the old loan's status to renewed.

Refer to the official CreditBook Microcheck API Documentation for the current request parameters, response structures, and error codes for every call in this flow.

docs.creditbook.ca/api

6. Reporting loan requests

6.1 Purpose

A Loan Request is your record of a customer's loan application, contributed to CreditBook's shared alternative-credit database so other participating lenders, and you, can see an applicant's history across contributors.

6.2 What makes a complete record

Minimum required by the API to create a Loan Request:

Field	Type	Notes
<code>requestId</code>	string	Your own loan ID. Must be unique per lender.
<code>firstName / lastName</code>	string	Applicant's name.
<code>email</code>	string	Must be a valid email address.
<code>dateOfBirth</code>	date	Applicant's date of birth.
<code>requestDate</code>	date-time	Date/time the loan was requested.

For a record that's actually useful to other contributors, also include: `address`, `phone1`, `job`, `references`, `amount`, `balance`, `ipAddress`, and `customerReferenceId`. `address`, `job`, and `references` are optional, but once you include one, its own sub-fields (e.g. `address` needs `city`, `country`, `stateProvince`, `streetAddressLine1`, `zipPostalCode`) become required.

6.3 Status

Status is the field other contributors rely on most. Keep it current via the status call. Defaults to new at creation. Main groups: New Loan (new, pre-approbation), Incomplete Application (missing-ibv, missing-information, missing-signed-contract, waiting-for-customer), Lender Decision (accepted, refused, request-canceled, duplicates), and Loan Journey (in-collection, in-payment-agreement, loan-pay-in-full, warning, bankruptcy, consumer-proposal, fraudster, renewed). A reason is required whenever status is set to refused.

6.4 Example: creating a Loan Request

The example below is illustrative. Field names and formats shown reflect the current schema, but always confirm against the official documentation before building.

```
POST /loanrequests
X-API-Key: YOUR-API-KEY-HERE
Content-Type: application/json

{
  "requestId": "JS0001112",
  "firstName": "John",
  "lastName": "Smith",
  "dateOfBirth": "1990-04-12",
  "email": "john.smith@example.com",
  "phone1": "(555) 555-5555",
  "requestDate": "2026-09-16T14:05:00.000Z",
  "amount": 500.00,
  "address": {
    "streetAddressLine1": "123 Example Street",
    "city": "Montreal", "country": "CA",
    "stateProvince": "QC", "zipPostalCode": "H0H 0H0"
  }
}
```

CreditBook returns the record as stored, including the generated id you'll need for every later call on this loan.

Refer to the official CreditBook Microcheck API Documentation for the current field-by-field schema, every status/reason definition, and the Search, Signal, and Request Tracker endpoints (a separate, optional integration for querying other contributors' data).

docs.creditbook.ca/api

7. Developer implementation guidance

7.1 Identifiers and references

- Store the CreditBook-generated id returned when a Loan Request is created against your own record. Every later call (update, status, revisions, view) is keyed on it.
- Keep your own requestId unique per lender. It's how you cross-reference the record on your side.
- Use referenceId to link a renewal to the loan it replaces (see Section 5).

7.2 Integration decisions to make up front

Decision	Options	What it affects
How you keep records current	Update each record individually as changes happen, or sync many at once on a schedule.	Whether you use the single-record update call or the batch update call, and how quickly the shared database reflects your data.
How you report status	Call the status endpoint as decisions happen, or batch status changes with your other updates.	How current your Loan Requests look to other contributors relying on status.
How you handle renewals	Link a new loan to its predecessor with <code>referenceId</code> , or treat it as an unrelated new record.	Whether the prior loan's status is automatically updated to renewed.

7.3 Sequencing example

The example below shows the order of operations for reporting a new loan application: create the record, store the id, then report status once a decision is made. Field names and endpoint paths are illustrative; take the current values from the online documentation.

```
// 1. Create the Loan Request and store the returned id
response = POST /loanrequests { requestId, firstName, lastName, ... }
creditbookId = response.id

// 2. Push updates as details change
PUT /loanrequests/{creditbookId} { balance, comments, ... }

// 3. Report a decision once one is made
POST /loanrequests/{creditbookId}/status { status: "accepted" }
```

Refer to the official CreditBook Microcheck API Documentation for the current endpoint paths and parameters used in this example.

docs.creditbook.ca/api

8. Testing and sandbox guidance

Sandbox Loan Requests are stored the same way as production, just isolated under your sandbox key and quota. There is no simulated third party to connect to, so test data is entirely up to you.

Test approach

- Run the full lifecycle end to end in sandbox: create, update, status change, and renewal.
- Exercise a refused status with each reason your business uses, not only accepted.

- Test the batch update endpoint if you plan to sync many records at once.
- Confirm your production API key and requestId mapping before go-live. Sandbox and production records are not interchangeable.

Refer to the official CreditBook Microcheck API Documentation for current rate limits and any sandbox-specific reference detail.

docs.creditbook.ca/api

9. Official API documentation

The official CreditBook Microcheck API Documentation holds the current reference detail for the API: request parameters, request and response structures, and status/reason codes. Use it wherever this guide points to it.

docs.creditbook.ca/api

For account setup, your API key, or anything about your Microcheck agreement, contact Inverité. For questions about CreditBook's API behavior itself, contact the CreditBook team directly.