myAGA Public API — Getting Started Guide

An introduction for financing partners (banks) who are new to myAGA and this API.

This guide is a companion to the technical documentation — it explains the "what" and "why" before you dive into the "how." For the technical contract and ready-to-use examples, see:

- **README** — how to run/reach the service and a working curl quickstart
- **OpenAPI specification** (`my-aga_applications_for_export_credit_cover-2.0-swagger.yaml`) — the full technical contract: every endpoint, field, and data type
- **Postman collection** (`myAGA API v2.postman_collection.json`) — ready-to-import example requests

Who This Guide Is For

You work for a financing partner — typically a bank — that wants to integrate with myAGA to exchange export credit cover application data. You may not yet know what myAGA is, what an "export credit cover application" contains, or how the pieces fit together. This guide gets you oriented before you open the technical documentation.

What Is myAGA?

myAGA is the customer portal operated by AGA (Euler Hermes) through which German exporters apply for export credit cover for their export transactions. An exporter fills in the details of a planned export — the goods or services, the importing country, payment terms, and so on — and submits this as an **export credit cover application**. AGA reviews and processes these applications.

Financing partners such as banks are often involved in the same transaction — for example, providing a buyer credit to finance the export. To do that, a bank typically needs the same underlying data the exporter already entered into myAGA, and today that usually means asking the exporter to send it separately, by email or otherwise, duplicating work that was already done in the portal.

What Does This API Do?

The myAGA Public API lets an authorized financing partner exchange export credit cover application data directly with the myAGA platform, instead of relying on the exporter to forward it manually. It is designed for **server-to-server, unattended use** — for example, a nightly batch job or a periodic poll — not for a human clicking through a UI.

There are two directions of use:

1. **Reading data.** Query applications that the exporter has already completed and submitted in myAGA, along with any attached documents (certificates, reports, project descriptions, etc.). This covers the case where the exporter has finished their application and now wants to add you as their financing partner, without re-entering everything.
2. **Pushing data.** Send a new application, pre-filled with data you already have, into the exporter's myAGA workspace as a **draft**. The exporter then reviews and completes it in the portal UI rather than starting from a blank form. This covers the case where you are helping the exporter get started.

Both directions use the same data structure, so an application looks the same regardless of which side created it.

Key Concepts

A few terms you'll see throughout the technical docs:

- **Exporter** — the German company applying for export credit cover. Identified in the API by a **personNo** (also called "Deckungsnehmer" identifier internally).
- **Export Coverage Application** — the core resource of this API: one export transaction's application data (parties involved, goods/services, payment terms, amounts, dates, and so on).
- **Submitted vs. Draft** — an application entered and finished by the exporter in the myAGA UI is "submitted." An application your bank pushes via this API starts as a "draft" until the exporter reviews and completes it. Only submitted applications appear when you query the collection of applications; drafts you pushed can still be retrieved directly by their id.
- **Attachments** — binary documents (certificates, reports, presentations, etc.) associated with an application. Attachment access can be restricted by document category depending on how the application was created.
- **Partner scoping** — you will only ever see applications that are associated with your organization. Which applications that is gets determined by AGA/the exporter on authorization; it is not something you specify in a request.

Getting Access

Access to this API is not self-service. Your organization must first be a certified financing partner recognized by AGA, and AGA provisions the credentials your systems will use to authenticate (a client id and client secret for the OAuth2 "client credentials" flow — see the README and OpenAPI spec for how to use them technically). If your organization is not yet set up this way, that is the first step — see **Getting Help** below.

Reference Data / Code Lists: Not Yet Available via the API

The technical documentation mentions standard code lists — for example ISO country codes, currency codes, and similar reference data used throughout an application's fields. **These code lists are not currently provided as part of this API, nor as a downloadable file, because AGA does not yet have a reliable way to retrieve and maintain them itself.**

In practice, this means: until a proper reference-data endpoint exists, please align on the exact codes and values you intend to use directly with AGA (see **Getting Help**) rather than assuming a specific external code list will be accepted as-is. This is a known gap, already tracked internally for a future improvement — it is not something you need to work around on your own.

Where to Find the Technical Details

Once you're oriented, the actual integration work happens in the technical documents:

Document	Use it for
README	Running/reaching the service, a minimal working curl example
OpenAPI spec (Swagger YAML)	The authoritative contract — every endpoint, parameter, and field
Postman collection	Hands-on example requests you can run once your credentials are set up

Getting Help

If you run into problems — access, credentials, unclear data, or anything else — please contact AGA using the contact information currently published on the myAGA website (my.agaportal.de). We intentionally don't name a specific person here, since the responsible contact can change over time; the website is kept up to date.