SESAME BANKHALL GROUP | JammJar

DELIVERABLE 02

OneView Integration Design

How JammJar connects to OneView to enable a centralised single customer view across SBG — three depths, a clear authentication path, and a 424-field data map across seven entities.

DATE	AUDIENCE	CLASSIFICATION
April 2026	SBG Executive	Confidential

READING MAP

Contents

Six cards. The first frames the architecture at the group level. The next three describe the three integration options — choices, not phases. The last two cover authentication and the data mapping.

00 OneView as the group oversight hub

OneView is SBG's single source of truth for activity across its PMS and Network businesses. JammJar's role is to feed it — light to deep, depending on the depth SBG chooses.

Today the PMS member base predominantly runs on two CRMs: Acre (which now sits with a competitor) and Smartr365 (which has no API). Both feature gaps mean OneView is starved of the rich, real-time adviser-level data SBG needs to do meaningful network oversight, AI-assisted compliance scoring or cross-firm reporting.

JammJar sits structurally on the inside of the PMS / DA segment — the upgrade path for firms that want to leave Acre, and the AI-native default for net-new members. As JammJar's footprint inside the member base grows, OneView gets a richer oversight surface for those firms; SBG's internal compliance and network teams gain reporting depth that doesn't exist today. The offering to PMS club members expands, and the quality of group oversight lifts in step.

01 Option 1 — Event-driven push & SSO

The lowest SBG-side lift. Events from JammJar push to OneView in near-real-time over a webhook with HMAC signing. SSO is OAuth 2.0 with OneView as IdP.

JammJar emits state-change events as they happen — `application.status_changed`, `check.completed`, `suitability.signed`, and a couple of dozen more. Each event has a versioned schema, an idempotency key per event, and is HMAC-signed using a per-tenant shared secret. OneView receives, acknowledges and maps events onto its data model.

SSO is OAuth 2.0 with PKCE — OneView issues the token, JammJar accepts it and authenticates the adviser inside JammJar. The adviser logs into OneView once; their JammJar session is established by the token. **Deliverable by JammJar alone against today's schema** — no SBG-side build is required beyond standing up the OneView OAuth IdP and the OneView event-receiver endpoint.

02 Option 2 — Bidirectional REST

Adds pull and drill-down on top of Option 1. OneView calls JammJar APIs for cases, advisers, firms, documents — visible inside the OneView UI without leaving it.

Option 2 opens up two-way traffic. Events still flow as in Option 1 — Option 2 adds REST endpoints on the JammJar side, scoped per OneView call, with OAuth-bearer auth. List, search, pagination, filtering, full case detail; OneView embeds JammJar widgets natively in OneView screens. For deeper investigation OneView can also use the JammJar OneView-key-bearing endpoint, which has cross-tenant aggregate access. **Requires SBG-side REST endpoints** for the equivalent OneView entities — firms, advisers, policies, commissions — so JammJar can resolve the OneView identity model when constructing responses.

03 Option 3 — Full MCP service layer

The agent-consumable surface. JammJar exposes its full capability set as MCP tools, callable directly from OneView's AI layer — natural language audit, fact-find completeness, deposit assessment, suitability generation.

MCP (Model Context Protocol) is the open agent-consumable layer Anthropic and others have standardised on. The JammJar MCP server exposes six headline tools — `run_file_audit`, `get_audit_status`, `check_fact_find_completeness`, `assess_deposit_sufficiency`, `generate_suitability_report`, `get_case_history` — each fully typed, each with consistent error semantics.

Behind the tools sits the same fact-find, audit and suitability infrastructure JammJar's own UI uses — there is no second-class MCP layer. An OneView AI assistant can answer compliance and oversight questions across the network by composing tool calls; SBG staff can use the same interface for their own queries. **Requires SBG-side MCP clients** to consume the tool catalogue — the JammJar side is shipping today.

04 Authentication — OneView as identity provider

OAuth 2.0 with PKCE. OneView issues the token. JammJar accepts it. Shane Wright (SBG CTO) confirmed this as the required pattern.

Every option above leans on the same authentication pattern: OneView is the identity provider, JammJar is the relying party. PKCE-secured OAuth 2.0 means no client secret leaks, refresh-token rotation, fine-grained scope per relying party. Adviser identity, firm identity and OneView-system identity are three distinct token classes — JammJar's authentication layer differentiates them by scope claims, not by separate flows. Tokens carry the OneView `firm_id` and `adviser_id` directly; JammJar maps these to its tenant-scoped customer / case identifiers via a join table populated when an adviser first signs in. **Confirmed by Shane Wright** as the SBG-side requirement before commercials.

05 Data mapping — 424 fields across seven entities

Customer, Case, Application, Fact-find, Documents, Audits & Checks, Activity & Telephony. Each field tagged with PII class, per-option availability, and whether it ships today (Now) or requires a Proposed build.

The mapping is the operational truth of the integration. Every field in the JammJar production schema is listed against every entity OneView cares about, with three pieces of metadata per field: PII class (which determines encryption, retention and audit-log behaviour), per-option availability (some fields require Option 2 or Option 3 depth to reach, e.g. cross-tenant rollups), and Now-vs-Proposed (some require new rollup tables, cross-tenant aggregation, or OneView-side data layers).

Where the integration depends on OneView-side joins — particularly `firm_id` and `network_id` — these are annotated inline. Proposed items are explicitly named and separated from Now items so SBG-side engineering can scope dependencies up front.