Livelink Integration High Level Design

Document Control

Author	Jiri Ludvik
Version	0.7
Date	17 Dec 2025
Status	DRAFT
Reviewer	Peter Hase, Amr Elshekh, Roddy Herries, James Jackson
Approver	Jiri Ludvik

Context

EBRD is seeking to enhance the integration capabilities between business applications. The initiatives that can benefit from these are as follows

- **AI Foundations:** Access to information managed in Livelink represent the most frequent business requirement for AI project. Open standard APIs are the most effective method of providing access
- **TIBCO Decommissioning:** TIBCO ESB is currently facilitation between various CSG systems and Livelink. As part of EBRD Data Brokerage strategy, TIBCO ESB is planned to be replaced by Azure AI Services and decommissioned by 2028. Functional equivalent of TIBCO Content Adapter endpoints are required to facilitate decommissioning of the system
- **Monarch:** Current method of integration with Livelink provided by TIBCO is inflexible and does not allow implementation of strategic needs of CSG Platform. Redesigned API functionality is required to support such needs
- **Livelink Metadata Improvement**: EBRD is seeking to improve metadata of documents in Livelink to improve its information management processes and to de-risk migration to Strategic Information Management System

Requirements

EBRD requires exposure of usable, secure, open-standard based APIs in Livelink

API Endpoint	**Priority**		**Benefiting Projects**			
		AI Project (Horizon 2)	**Donor Funds**	**Limits**	**TIBCO ESB Decom.**	**SIMS Migration**

List documents in a folder	1	M	M	O		O
Search	1	M				O
Read metadata (incl. URL)	1	M		M	M	M
Read document	1	M	M			O
Update document metadata	1	O				O
Rename object	1	O			M	M
Create document (with metadata)	2			O	M	M
Create submission	2				M	M
Create folder	2				M	M
Process file batch	2				M	

Key:

M - Mandatory

O - Optional

Non-Functional Requirements

- APIs must enforce end-user authorisation, using Oauth2 protocol.
- APIs must minimise the rework to consumers following implementation of SIMS
- APIs must be easy to use for developers

Current State

Open Text Content Server offers a range of native REST APIs; however, these involve a number of challenges:

1. They expose system's internal data structure known as the "DTree", in which every system object is represented as an abstract Node of this tree (with subtypes distinguishing whether a node is a folder, document or other object);
2. They do not enforce pre-defined business rules in regards to creation of the documents and associated permissions
3. They manage metadata (using an abstract called Category) as a data entity separate from Node. A category has its own schema, and can be applied to to one or many nodes. A single node can have multiple categories applied to it;

4. Developer must traverse Category and Node schemas to retrieve foreign keys required to update metadata;
5. Categories are hierarchical concept with inheritance rules, complicating metadata updates;
6. Search endpoint uses proprietary search query syntax;
7. Developer documentation is insufficient for complex applications;
8. Content Server authorisation relies on proprietary OpenText authentication scheme, complicating the integration flows, and requiring developers who are not specialists in Opentext to understand their proprietary technology.

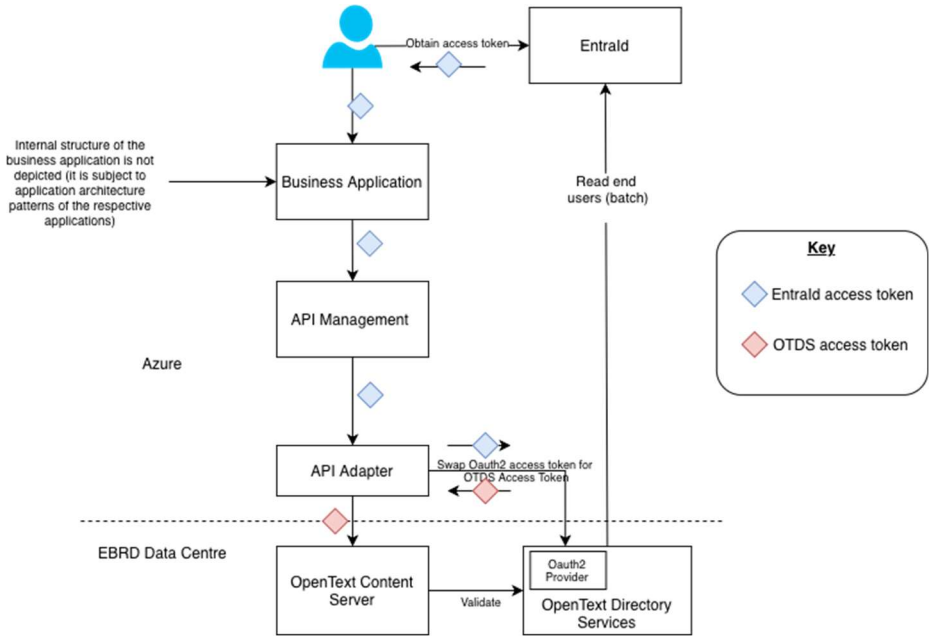
Solution

To minimise the impact of SIMS future migration on consumers, consumers will access Content Server APIs via the means of custom adapter(s) protecting them from internal ContentServer technical realisation.

Key Design Decisions applicable to technical realisation of the adapter(s) are as follows:

- Adapter Public API User Authentication: Oauth2 PKCE (design to support different applications TBC)
- Content Server API Authentication: OpenText Directory Services Issued access token
- Adapter programming language: [Amr ElshekhTBC]
- Adapter execution environment: Azure Functions

An indicative solution architecture illustrating the overall solution is as follows.



1. User logs in to Entra Id via their Business Application

2. Entra Id provides user with an Identity token and Oauth2 Access Token for their business application
3. When the application requires to interact with Livelink, it calls the REST APIs exposed on Livelink Adapter passing the Oauth2 Access Token in the Authorization header.
4. (Optional) Livelink Adapter validates the incoming access token
5. Livelink Adapter authenticates itself to OpenText Directory Services (OTDS) using its Client ID and a secret
6. Livelink Adapter calls OTDS REST endpoint to swap EntraId provided Access Token for OTDS provided Bearer Token (using RFC9693 Token Exchange)
7. OpenText Directory Services receive Exchange request, validate Oayth2 Access Token using EntraId public keys and issues OTDS Access Token
8. Livelink Adapter translates business concepts (documents, folders etc) to internal Open Text Server concepts (nodes, categories), applying pre-defined business logic
9. Livelink API Adapter calls OpenText Content Server REST APIs using OTDS Bearer Token injected in the header.
10. Open Text Content Server accepts the requests with OTDS Bearer token, and performs internal action applying ACLs appropriate for the requesting end user

---