

Connexion de Power Query à DOORS 9 en lecture seule

Synthèse exécutive

Une connexion *directe* de Power Query à la « base » DOORS 9 (DOORS Classic) au sens SQL/ODBC/JDBC n'est généralement pas réaliste, car DOORS Classic s'appuie sur un serveur propriétaire avec une base intégrée (ce n'est pas une base relationnelle « commerciale » exposée en SQL). ¹ Dans l'architecture IBM, l'accès *supporté* se fait via les API et services de DOORS : DXL (scripting) côté client/serveur, et OSLC (via DOORS Web Access, DWA) pour l'accès HTTP/REST (OSLC RM v1/v2). ²

En conséquence, les solutions pratiques pour Power Query (Excel/Power BI) se rangent en deux familles :

1. **La plus fiable et la plus simple à industrialiser (recommandée) : export automatisé via DXL en batch** vers un **fichier plat (CSV/JSON)** sur un partage, puis ingestion par Power Query (fichier). DOORS sait exécuter un programme DXL en batch sans GUI (switch `-batch`), avec authentification et sélection de base/serveur via la ligne de commande. ³
2. **La plus « directe » côté Power Query (mais plus complexe) : exposer DOORS via DWA/OSLC (REST), mais les requêtes programmatiques OSLC sont protégées par OAuth 1.0a** (obligatoire) et ne sont pas accessibles sans jeton, ce qui dépasse les modes d'authentification standards du connecteur Web de Power Query. ⁴ La voie la plus robuste consiste alors à insérer un **proxy interne** (microservice) qui gère OAuth 1.0a et renvoie du JSON « BI-friendly » à Power Query via clé API/Basic.

Le meilleur compromis « fiabilité × facilité » dans la plupart des organisations est **DXL batch → CSV/JSON → Power Query** (rafraîchissement planifié). La meilleure option si vous avez un besoin quasi temps réel et une équipe capable d'opérer un composant supplémentaire est **DWA/OSLC + proxy**. ⁵

Référentiels et surfaces d'intégration disponibles dans DOORS 9

Hypothèses reprises et explicitées

Vous êtes sur DOORS 9 « classic » on-prem, avec identifiants en lecture seule et connectivité réseau vers le serveur DOORS. DOORS Classic fonctionne avec un **serveur propriétaire et une base intégrée**, et peut être étendu/intégré via **DXL** et APIs. ¹

DOORS Web Access (DWA) est un composant optionnel qui apporte l'accès web et sert de socle aux interfaces OSLC RM v1/v2. ⁶

Implication clé pour ODBC/JDBC

Le fait que DOORS Classic soit décrit comme un « proprietary server with built-in database » indique qu'il n'expose pas nativement un modèle SQL/ODBC/JDBC comparable à une base Oracle/SQL Server/

PostgreSQL. ¹ En pratique, **ODBC/JDBC ne deviennent viables qu'avec une couche intermédiaire** (entrepôt de reporting, driver « XML ODBC » historique, staging) — détaillé plus loin.

Surfaces d'accès pertinentes pour Power Query

- **DXL en batch (offline/planifié)** : DOORS peut démarrer sans UI, exécuter un script DXL, puis s'arrêter (`-batch`). En batch, on fournit typiquement `-user`, `-password`, `-project` et la cible `-data port@server`. ⁷
- **OSLC RM via DWA (online/REST)** : DOORS agit comme fournisseur OSLC RM v1/v2, avec opérations GET/PUT/QUERY (notamment QUERY au niveau module en RM v2). ⁸
- **OSLC DXL Services (REST + exécution de DXL)** : DOORS peut exposer des scripts DXL exécutables via HTTP/HTTPS (« OSLC DXL services »), ce qui permet de renvoyer des payloads plus simples (CSV/JSON) qu'un flux RDF OSLC « pur ». ⁹

Comparaison des approches et arbre de décision

Tableau comparatif

Approche	"Direct" Power Query ?	Prérequis principaux	AuthN/AuthZ	Performance	Effort de mise en œuvre / maintenance	Points forts
DXL batch → CSV/JSON → Power Query (fichier)	Indirect (via fichier)	Client DOORS sur Windows, accès réseau, planification	Identifiants DOORS côté job batch ; côté PQ typiquement Windows/partage	Très bonne en lecture (fichier), scalable via découpage	Faible à moyen (1 script + un job + M query)	Très robuste, simple à auditer, découple BI de DOORS
DWA OSLC RM v2 → Power Query (Web)	Oui (HTTP) mais...	DWA opérationnel + OSLC configuré	OAuth 1.0a obligatoire pour requêtes programmatiques	Variable ; dépend requêtes OSLC et pagination	Élevé si fait "à la main" dans M	Accès online, modèle standard OSLC
DWA OSLC + Proxy interne → Power Query (Web API)	Oui (HTTP proxy)	DWA + proxy (Python/.NET...), monitoring	Proxy gère OAuth 1.0a côté DOORS ; PQ utilise API key/Basic	Bonne si proxy met en cache / agrège	Moyen (un service à opérer)	Le plus "direct" et industrialisable pour BI
OSLC DXL Services → Proxy/PQ	Oui (HTTP) mais OAuth	DWA + OSLC DXL services	OAuth 1.0a (programm.); DXL service admin	Très good si DXL renvoie JSON ciblé	Moyen à élevé	Contrôle total du payload (JSON plat)

Approche	"Direct" Power Query ?	Prérequis principaux	AuthN/AuthZ	Performance	Effort de mise en œuvre / maintenance	Points forts
ODBC "historique" via XML ODBC driver + XDC	Oui (ODBC)	Installation driver XML ODBC + XDC + DWA	Dépend config (souvent basé sur REST/XML + mapping)	Souvent correct, mais dépend outil ancien	Élevé et peu recommandé	Rend l'accès "SQL-like"
Entrepôt/ reporting externe (DB2/SQL Server/ Oracle...)	Oui (ODBC natif)	ETL/collecte DOORS → DB relationnelle	Selon DBMS (AD, LDAP, etc.)	Excellente pour BI	Élevé	BI à grande échelle, requêtes SQL

Les trois constats structurants derrière ce tableau : (1) DOORS Classic est propriétaire avec base intégrée. ¹ (2) DWA publie OSLC RM v1/v2, mais les accès programmatiques sont protégés via OAuth 1.0a et inaccessibles sans jeton. ¹⁰ (3) Power Query (connecteur Web standard) expose des modes d'authentification tels que Anonymous/Windows/Basic/Web API/Organizational account, mais pas un flux OAuth 1.0a prêt à l'emploi, ce qui motive proxy ou connecteur custom. ¹¹

Étapes de détection quand l'infrastructure exacte est inconnue

1. **Confirmer le périmètre** : DOORS « classic » (desktop) vs DOORS Next. La doc IBM oppose explicitement DOORS (serveur propriétaire + base intégrée) à DOORS Next (Jazz Team Server + base commerciale), ce qui impacte totalement le "SQL access". ⁶
2. **Vérifier si DWA est présent** : tester l'URL racine et surtout le point d'entrée OSLC **statique** `/dwa/public/rootservices` (toutes les autres URLs OSLC sont dérivées et peuvent changer). ¹²
3. **Vérifier la configuration DOORS↔DWA** : OSLC RM v1/v2 nécessite que la configuration dbadmin inclue `-dwaHost`, `-dwaPort`, `-dwaProtocol`. IBM fournit un format de commande dbadmin type incluant ces paramètres. ¹³
4. **Identifier le port DOORS** : la ligne de commande/client DOORS utilise `-data port@server`. Exemple de doc IBM montrant `36677@myserver`. ¹⁴ Une documentation d'installation historique indique 36677 comme port par défaut. ¹⁵
5. **Si vous cherchez un DBMS "derrière"** : pour DOORS Classic seul, partez du principe qu'il n'y en a pas. Un DBMS n'entre en jeu que si vous avez déployé un **entrepôt** ou un outillage de reporting externe.

Solution recommandée : export DXL automatisé vers fichiers (CSV/JSON) + Power Query

Cette solution est la plus simple à sécuriser en lecture seule et la plus fiable en exploitation BI, parce qu'elle évite OAuth 1.0a et n'exige pas DWA. Elle exploite la capacité native de DOORS à exécuter DXL en batch et à se connecter au serveur DOORS via `-data port@server`. ³

Pré-requis logiciels et configuration

- **Machine d'exécution** : Windows (le client DOORS fonctionne uniquement sur Windows, et à partir de 9.6.1 il est fourni en client 64-bit). ¹⁶
- **Client DOORS installé** sur la machine qui exécutera le job batch (souvent un serveur d'intégration/ETL interne).
- **Accès réseau** au serveur DOORS et au port DOORS (via `-data port@server`). ¹⁴
- **Compte DOORS lecture seule** (ou à minima sans droits d'écriture sur les modules ciblés). La sécurité/les droits d'accès se configurent au niveau base/projets dans les propriétés de base. ¹⁷
- **Emplacement de sortie** : partage réseau en lecture pour les consommateurs BI (Excel/Power BI).
- **Planification** : Planificateur de tâches Windows (ou orchestrateur d'entreprise) pour lancer l'export à intervalle fixe.

Étapes pas à pas côté DOORS (DXL + batch)

Étape 1 : définir le périmètre d'export

Décidez si vous exportez : (a) une liste de modules, (b) un dossier/projet entier, (c) un module et une vue particulière. La doc IBM rappelle que DOORS supporte les exports « document/spreadsheet » et qu'il existe un langage DXL pour automatisation, intégration et import/export. ¹

Étape 2 : écrire un script DXL d'export "read-only"

Ci-dessous un **exemple** minimaliste orienté BI : export CSV UTF-8 avec des colonnes stables (module, identifiants objet). Adaptez selon vos attributs DOORS (nom exact des attributs, vues, DXL disponible).

```
// export_module_to_csv.dxl (exemple)
// Paramètres via variables d'environnement (évite de hardcoder dans DXL)
string outFile = getenv("DOORS_EXPORT_FILE")
string modulePath = getenv("DOORS_MODULE")

if (null outFile || outFile == "" || null modulePath || modulePath == "") {
    print "Missing DOORS_EXPORT_FILE or DOORS_MODULE\n"
    halt
}

Module m = read(modulePath, false) // ouverture en lecture
if (null m) { print "Cannot open module: " modulePath "\n"; halt }

Stream out = write(outFile)
out << "module_path;object_abs_no;heading;text\n"

Object o
for o in m do {
    // Remplacer "Object Heading" / "Object Text" par vos attributs ou DXL
    columns
    string h = richText(o."Object Heading")
    string t = richText(o."Object Text")
    out << modulePath ";" (string) (absNo o) ";" h ";" t "\n"
}

close(m)
```

```
close(out)
print "OK\n"
```

Points de conception BI importants : (1) exportez un identifiant stable (ex. chemin module + `absNo`), (2) sortez les attributs clés (statut, priorité, baseline, etc.), (3) si le volume est élevé, préférez un export "delta" (par date de modification) ou un export partitionné (1 fichier/module). La capacité d'exécuter DXL en batch et de piloter le mode d'ouverture (READ_ONLY) est documentée côté client DOORS. ⁷

Étape 3 : exécuter DOORS en batch

IBM documente le switch `-batch (-b)` : DOORS démarre sans GUI, exécute le programme DXL, puis s'arrête. En batch, on fournit d'ordinaire `-user`, `-password`, `-project` et la cible `-data port@server`. ⁷

Exemple de commande (à adapter) :

```
set DOORS_EXPORT_FILE=\\fileserver\doors_exports\reqs_moduleA.csv
set DOORS_MODULE=/Projet/Exigences/ModuleA

"C:\Program Files\IBM\Rational\DOORS\9.x\bin\doors.exe" ^
-data 36677@doors-dbserver ^
-u "readonly_user" ^
-p "*****" ^
-p "MonProjet" ^
-defopenmode READ_ONLY ^
-b "C:\doors_scripts\export_module_to_csv.dxl" ^
-W nowait
```

Références utiles dans la doc IBM :

- `-data` prend `port@server` (ex. `36677@myserver`). ¹⁴
- `-defopenmode READ_ONLY` permet de définir le mode d'ouverture par défaut. ⁷
- `-W nowait` ferme les fenêtres de sortie en batch. ⁷

Étape 4 : planifier et journaliser

Industrialisez avec : (a) un compte de service Windows dédié, (b) logs et rétention (gardez N derniers exports), (c) contrôle d'erreur (taille nulle, "OK" attendu). Côté sécurité, notez que certaines options client (ex. streaming/caching) peuvent temporairement mettre des données en cache local, ce qu'IBM signale comme risque potentiel. ⁷

Étapes pas à pas côté Power Query

Étape 1 : se connecter au CSV (ou JSON) exporté

Dans Excel/Power BI : *Obtenir des données* → *Texte/CSV* (ou *Depuis un dossier* si vous exportez un fichier par module).

Exemple Power Query M (CSV sur partage réseau)

```
let
    Source = Csv.Document(
```

```

        File.Contents("\\fileserver\\doors_exports\\reqs_moduleA.csv"),
        [Delimiter=";", Encoding=65001, QuoteStyle=QuoteStyle.Csv]
    ),
    Promoted = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    Typed = Table.TransformColumnTypes(
        Promoted,
        {
            {"module_path", type text},
            {"object_abs_no", Int64.Type},
            {"heading", type text},
            {"text", type text}
        }
    )
in
    Typed

```

Exemple Power Query M (JSON exporté)

Si vous préférez produire du JSON (souvent plus simple pour données hiérarchiques), Power Query peut consommer via `Json.Document` :

```

let
    Raw = Json.Document(File.Contents("\\fileserver\\doors_exports\\reqs_moduleA.json")),
    Items = Raw[items],
    AsTable = Table.FromRecords(Items)
in
    AsTable

```

Considérations de sécurité pour cette solution

- **Principe du moindre privilège** : le compte DOORS utilisé par le batch doit être strictement lecture seule (droits d'accès DOORS configurables au niveau base et hérités). ¹⁷
- **Secrets** : évitez d'inscrire le mot de passe DOORS en clair dans un fichier partagé. Favorisez l'exécution du batch sur un serveur contrôlé (ACL NTFS, compte de service, secrets gérés par l'ordonnanceur).
- **Données au repos** : chiffrez le partage (au minimum ACL + chiffrement disque), surtout si vous exportez du texte d'exigences sensible.
- **Traçabilité** : conservez un log d'exécution et la version du script DXL utilisé.

Schéma de flux pour la solution recommandée

```

flowchart LR
    A[Serveur DOORS 9 (base intégrée)] -->|Protocole DOORS client/server| B[Client DOORS en batch (-batch DXL)]
    B -->|CSV/JSON| C[Partage fichiers sécurisé (SMB)]
    C -->|File.Contents / Csv.Document / Json.Document| D[Power Query (Excel/Power BI)]
    D --> E[Modèle BI / tableaux / rapports]

```

Alternative plus « directe » : DWA OSLC / OSLC DXL Services via un proxy REST

Cette option vise un rafraîchissement plus fréquent (voire proche temps réel) sans passer par des exports fichiers, mais elle devient rapidement un sujet **d'authentification OAuth 1.0a** et de robustesse des URLs OSLC.

Points structurants à connaître avant de commencer

- **Point d'entrée OSLC stable** : `/dwa/public/rootservices` est le seul URL statique publié ; tous les autres sont dérivés et susceptibles de changer (donc on évite de hardcoder des chemins internes). ¹²
- **OAuth 1.0a obligatoire** pour requêtes programmatiques OSLC RM v1/v2, et **impossible d'accéder à une ressource protégée sans jeton**. ¹⁸
- Le connecteur Web Power Query propose des modes d'authentification (Anonymous/Windows/Basic/Web API/Organizational account) mais pas un flux OAuth 1.0a "direct" prêt à l'emploi. ¹⁹

L'option "propre" consiste donc à mettre un **proxy interne** qui : 1) gère OAuth 1.0a vers DWA/OSLC, 2) transforme en JSON tabulaire stable, 3) expose une API interne simple (clé API) consommable par Power Query.

Pré-requis côté DOORS/DWA

Étape 1 : installer/valider DWA et SSL/TLS

IBM décrit DWA comme composant optionnel de DOORS Classic, et fournit une procédure de configuration SSL/TLS (certificat + keystore) pour sécuriser l'accès web. ²⁰

Étape 2 : configurer la communication DOORS DB server ↔ DWA (dbadmin)

IBM fournit un format de commande `dbadmin` incluant `-dwaHost`, `-dwaPort`, `-dwaProtocol` (et des paramètres DCN/broker). ²¹

Étape 3 : récupérer Consumer Key + Secret

IBM indique qu'il faut fournir au consommateur (application tierce) un Consumer Key, un OAuth Secret et un Root Services URL, et explique où lire `consumer_key` / `secret` via *File > OSLC > Local keys* (DOORS/DWA). ²²

Étape 4 : choisir OSLC "pur" vs OSLC DXL services

- OSLC "pur" est standard mais renvoie souvent RDF/XML et implique de gérer la découverte/pagination/queries. DOORS supporte notamment QUERY au niveau module en RM v2. ⁸

- **OSLC DXL services** permet d'exécuter un DXL via HTTP/HTTPS ; IBM donne un exemple d'URI `/dwa/rm/dxl/helloWorld` et documente l'installation (outil disponible à partir de DOORS 9.6.1.3). ²³

Pour un usage BI, OSLC DXL services est souvent plus efficace, car vous contrôlez le payload (JSON "plat") et limitez les appels.

Mise en œuvre d'un proxy REST interne (recommandé pour OSLC)

Logiciels requis

- Un runtime pour microservice (Python/.NET/Node) + bibliothèque OAuth 1.0a (ex. Python `requests-oauthlib`).
- Connectivité HTTP(S) vers DWA.

Étape 1 : implémenter l'acquisition de jeton OAuth 1.0a

Les endpoints exacts sont visibles dans des exemples communautaires et outillages (ex. `.../dwa/oauth-request-token`, `.../dwa/oauth-access-token`, `.../dwa/oauth-authorize-token`).

²⁴ IBM confirme surtout le cadre : OSLC programmatique = OAuth 1.0a obligatoire. ²⁵

Étape 2 : appeler un endpoint de données DOORS

Deux stratégies : - Appeler le catalogue OSLC (ex. `/dwa/rm/discovery/catalog` cité comme ressource protégée) puis suivre la découverte. ²⁶

- Appeler un **DXL service** dédié qui renvoie directement `application/json`.

Étape 3 : exposer une API interne "Power Query friendly"

Exposez un endpoint interne tel que `/api/doors/export/moduleA` renvoyant du JSON tabulaire, protégé par clé API.

Exemple Power Query M (consommation du proxy avec clé API)

Le connecteur Web de Power Query supporte l'authentification "Web API" (clé API) et l'ajout d'en-têtes dans une URL avancée. ²⁷

```
let
    Source = Json.Document(
        Web.Contents(
            "https://doors-proxy.intra.local/api/doors/moduleA",
            [
                Headers = [
                    #"x-api-key" = "*****",
                    Accept = "application/json"
                ]
            ]
        )
    ),
    Items = Source[items],
    Table = Table.FromRecords(Items)
in
    Table
```

Sécurité

- Les appels OSLC programmatiques sont conçus pour être inaccessibles sans jeton OAuth (ce qui est un point de sécurité majeur). ²⁵
- Centraliser la gestion des secrets (consumer key/secret DOORS) dans le proxy réduit le risque de fuite dans des fichiers Excel/Power BI.
- Sur DWA, maîtrisez la durée de vie des jetons OAuth (paramètres côté configuration DWA/festival.xml mentionnés dans la doc de dépannage). ²⁸
- Si vous utilisez OSLC DXL services, IBM note que l'administrateur peut verrouiller l'accès à certaines fonctions pour réduire les risques. ²⁹

Approches ODBC/JDBC et autres variantes (entrepôt, driver XML ODBC, COM) et sécurité

ODBC “direct” vers DOORS : pourquoi ce n’est pas l’option prioritaire

DOORS Classic est présenté par IBM comme un produit à serveur propriétaire et base intégrée, étendu via DXL/APIs, ce qui ne correspond pas à une exposition SQL standard. ¹ Pour Power Query, cela signifie : **pas de DSN ODBC “DOORS database” officiel** à brancher comme on le ferait avec SQL Server/Oracle, sauf via une couche qui transforme/republie les données.

Variante ODBC “indirecte” via un driver XML ODBC + XDC (historique Rational Insight)

Il existe historiquement une approche où un **driver ODBC “XML”** transforme des appels ODBC en requêtes REST/GET et reconvertit des documents XML en résultats relationnels, piloté par un fichier **XDC (XML Data Configuration)**. ³⁰ Des guides IBM Rational Insight montrent la création d’un DSN en choisissant « IBM Rational Insight XML ODBC Driver » et en pointant vers un fichier XDC. ³¹ Dans la communauté, il est mentionné qu’un `doors.xdc` peut être obtenu depuis un serveur DWA, puis enregistré dans unixODBC pour obtenir une interface ODBC vers DWA. ³²

Pourquoi ce n’est généralement pas recommandé en 2026 : des composants Rational Insight associés ont des statuts de support retraités (ex. retrait du support annoncé pour Rational Insight 1.1.x au 30/09/2018). ³³ En pratique, cette voie n’est à considérer que si votre organisation possède déjà cet héritage et accepte les risques de maintien.

Si vous devez tout de même l’utiliser (pas à pas, synthétique)

- 1) Installer les composants nécessaires (driver XML ODBC + outil XDC). ³¹
- 2) Obtenir/configurer le XDC DOORS (référence communautaire `doors.xdc`). ³²
- 3) Créer un DSN ODBC pointant vers le XDC (les guides IBM montrent ce pattern DSN + XDC). ³¹
- 4) Dans Power Query : *Obtenir des données* → *ODBC*, sélectionner le DSN ou fournir une chaîne `dsn=...`. ³⁴
- 5) Utiliser `Odbc.DataSource` ou `Odbc.Query` en M selon que vous voulez naviguer ou pousser une requête SQL. ³⁵

JDBC : limites et contournements

Power Query expose des fonctions et connecteurs orientés ODBC/OLE DB/OData/Web plutôt qu’un connecteur JDBC générique. La liste des fonctions d’accès aux données documentées met en avant `Odbc.*`, `OleDb.*`, `Web.*`, etc. ³⁶ En pratique, si votre “cible” n’expose que JDBC, la stratégie la plus simple est de **trouver un pilote ODBC officiel du DBMS** ou de mettre un service intermédiaire. (C’est précisément ce qu’on recommande déjà pour DOORS : une couche intermédiaire ou export.)

Si vous avez (ou créez) un entrepôt relationnel (détection DBMS + alternatives)

Si votre organisation décide de stage DOORS dans une base relationnelle (ou si elle en a déjà une), Power Query via ODBC devient la voie classique.

Détection rapide du DBMS (si inconnu)

- Recherchez une URL JDBC dans la config de l’ETL/outil :

- jdbc:sqlserver://... → SQL Server
- jdbc:oracle:thin:@... → Oracle
- jdbc:db2://... → Db2
- jdbc:postgresql://... → PostgreSQL
- Côté Windows, vérifiez les pilotes installés dans l'administrateur ODBC (attention aux versions 32/64 bits). Microsoft documente l'existence de deux administrateurs ODBC et la séparation des DSN 32/64.

37

Chaînes ODBC typiques (exemples)

Power Query accepte une chaîne de connexion dans `Odbc.DataSource(connectionString, options)`. 38

```
// SQL Server (exemple générique)
Odbc.DataSource("Driver={ODBC Driver 18 for SQL
Server};Server=sql01;Database=DoorsStaging;Encrypt=yes;TrustServerCertificate=no;")
```

```
// PostgreSQL (exemple générique)
Odbc.DataSource("Driver={PostgreSQL
Unicode(x64)};Server=pg01;Port=5432;Database=DoorsStaging;")
```

Exemple `Odbc.Query` (SQL poussé)

`Odbc.Query` exécute une requête SQL native via un pilote ODBC. 39

```
let
    Source = Odbc.Query(
        "dsn=DoorsStaging",
        "select module_path, object_id, status, last_modified from req_objects
where status <> 'Deleted';"
    )
in
    Source
```

Sécurité et durcissement – points transverses (à appliquer quelle que soit l'approche)

- **DWA / OSLC** : OAuth 1.0a est imposé sur les requêtes programmatiques, et l'accès à une ressource protégée est impossible sans jeton. C'est un garde-fou de sécurité majeur, mais complexifie l'intégration Power Query "directe". 18
- **Chiffrement des flux** : privilégiez HTTPS pour DWA (IBM décrit la mise en place SSL/TLS via certificat/keystore). 40
- **Durcissement du serveur DOORS** : IBM documente des mécanismes de sécurité serveur (secure connections, interoperation servers, allowlist, certificats) et insiste sur l'usage d'une allowlist pour les interoperation servers. 41
- **Secrets dans Power Query** : Microsoft recommande d'éviter de hardcoder des clés API dans le code M et d'utiliser les mécanismes d'identifiants (Web API credential) quand possible. 42
- **ODBC** : attention aux environnements 32/64 bits (drivers + DSN) ; Microsoft explique la séparation des DSN et les emplacements des administrateurs ODBC. 37

Recommandation finale (priorisée)

- 1) **DXL batch → CSV/JSON → Power Query** (meilleur ratio fiabilité / simplicité, et très bonne maîtrise “read-only”). ³
- 2) **DWA OSLC + proxy interne** si besoin d'accès plus direct et fréquent, en acceptant l'implémentation OAuth 1.0a côté proxy. ⁴³
- 3) **ODBC “XML ODBC + XDC”** uniquement si héritage existant et contraintes fortes, en tenant compte des risques de support/outillage ancien. ⁴⁴
- 4) **Entrepôt relationnel** si vous avez des besoins BI à grande échelle (historisation, croisement multi-sources, gouvernance), mais c'est un projet d'architecture plus lourd.

Dans tous les cas, la coordination avec vos administrateurs DOORS/DWA est incontournable pour les paramètres dbadmin et clés OSLC (consumer key/secret). ⁴⁵

Enfin, dans l'écosystème Microsoft ⁴⁶ (Excel/Power BI), l'option “Web” de Power Query reste la plus simple pour consommer des APIs internes, tandis que l'option “ODBC” excelle dès qu'un stockage relationnel standard est disponible. ⁴⁷ Pour tout ce qui touche aux interfaces DOORS officielles (DXL/OSLC/DWA), les références principales viennent de IBM ⁴⁸ . ⁴⁹

¹ ² ⁶ ²⁰ ⁴⁹ <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors-next/7.1.0?topic=overview-comparison-doors-doors-next>

<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors-next/7.1.0?topic=overview-comparison-doors-doors-next>

³ ⁵ ⁷ ¹⁴ ⁴⁶ <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.2?topic=client-command-line-switches-doors-interoperation-server>

<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.2?topic=client-command-line-switches-doors-interoperation-server>

⁴ ¹⁸ ²⁵ ²⁶ ⁴³ ⁴⁸ <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=oslc-accessing-protected-resources>

<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=oslc-accessing-protected-resources>

⁸ https://www.ibm.com/docs/SSYQBZ_9.7.1/com.ibm.doors.install.doc/topics/r_doors_provider.html

https://www.ibm.com/docs/SSYQBZ_9.7.1/com.ibm.doors.install.doc/topics/r_doors_provider.html

⁹ ²³ ²⁹ <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=services-oslc-dxl-doors>

<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=services-oslc-dxl-doors>

¹⁰ ¹² https://www.ibm.com/docs/SSYQBZ_9.7.2/com.ibm.rational.dwa.install.doc/topics/c_troubleshootingOSLC.html

https://www.ibm.com/docs/SSYQBZ_9.7.2/com.ibm.rational.dwa.install.doc/topics/c_troubleshootingOSLC.html

¹¹ ¹⁹ ²⁷ ⁴⁷ <https://learn.microsoft.com/en-us/power-query/connectors/web/web>

<https://learn.microsoft.com/en-us/power-query/connectors/web/web>

¹³ ²¹ ⁴⁵ <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.1?topic=server-running-dbadmin-command>

<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.1?topic=server-running-dbadmin-command>

- 15 https://jazz.net/doors-admin/resource/attachment_14492054_doors_install_guide.pdf
https://jazz.net/doors-admin/resource/attachment_14492054_doors_install_guide.pdf
- 16 <https://www.ibm.com/docs/fr/engineering-lifecycle-management-suite/doors/9.7.2?topic=installing-doors>
<https://www.ibm.com/docs/fr/engineering-lifecycle-management-suite/doors/9.7.2?topic=installing-doors>
- 17 <https://www.ibm.com/docs/fr/engineering-lifecycle-management-suite/doors/9.7.1?topic=administering-configuring-database>
<https://www.ibm.com/docs/fr/engineering-lifecycle-management-suite/doors/9.7.1?topic=administering-configuring-database>
- 22 <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/lifecycle-optimization-publishing/7.0.3?topic=integration-configuring-oauth-oslc-authentication>
<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/lifecycle-optimization-publishing/7.0.3?topic=integration-configuring-oauth-oslc-authentication>
- 24 <https://jazz.net/forum/questions/151942/connect-to-doors-doors-web-access-via-python>
<https://jazz.net/forum/questions/151942/connect-to-doors-doors-web-access-via-python>
- 28 https://www.ibm.com/docs/SSYQBZ_9.7.1/com.ibm.rational.dwa.install.doc/topics/c_troubleshootingfestival.html
https://www.ibm.com/docs/SSYQBZ_9.7.1/com.ibm.rational.dwa.install.doc/topics/c_troubleshootingfestival.html
- 30 https://jazz.net/help-dev/rational-insight/topic/com.ibm.rational.raer.integration.doc/topics/t_integrate_data_source_rtc_xdc.html
https://jazz.net/help-dev/rational-insight/topic/com.ibm.rational.raer.integration.doc/topics/t_integrate_data_source_rtc_xdc.html
- 31 44 https://download.boulder.ibm.com/ibmdl/pub/software/rationalsdp/documentation/product_doc/System_Architect/pdf11312/English/RI_RSA_Integration_Guide.pdf
https://download.boulder.ibm.com/ibmdl/pub/software/rationalsdp/documentation/product_doc/System_Architect/pdf11312/English/RI_RSA_Integration_Guide.pdf
- 32 <https://jazz.net/forum/questions/243968/what-is-the-best-way-to-access-data-from-doors-96-for-analysis>
<https://jazz.net/forum/questions/243968/what-is-the-best-way-to-access-data-from-doors-96-for-analysis>
- 33 <https://www.ibm.com/support/pages/rational-insight11x-withdrawal-notification>
<https://www.ibm.com/support/pages/rational-insight11x-withdrawal-notification>
- 34 <https://learn.microsoft.com/en-us/power-query/connectors/odbc>
<https://learn.microsoft.com/en-us/power-query/connectors/odbc>
- 35 38 <https://learn.microsoft.com/en-us/powerquery-m/odbc-datasource>
<https://learn.microsoft.com/en-us/powerquery-m/odbc-datasource>
- 36 <https://learn.microsoft.com/fr-fr/powerquery-m/accessing-data-functions>
<https://learn.microsoft.com/fr-fr/powerquery-m/accessing-data-functions>
- 37 <https://learn.microsoft.com/en-us/troubleshoot/sql/connect/odbc-tool-displays-32-bit-64-bit>
<https://learn.microsoft.com/en-us/troubleshoot/sql/connect/odbc-tool-displays-32-bit-64-bit>
- 39 <https://learn.microsoft.com/fr-fr/powerquery-m/odbc-query>
<https://learn.microsoft.com/fr-fr/powerquery-m/odbc-query>
- 40 <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=dwa-configuring-ssl-tls>
<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=dwa-configuring-ssl-tls>

41 <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=security-configuring-secure-connection>

<https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.0?topic=security-configuring-secure-connection>

42 <https://learn.microsoft.com/en-us/powerquery-m/web-contents>

<https://learn.microsoft.com/en-us/powerquery-m/web-contents>