DEVELOPMENT ARCHITECTURE
WHITE PAPER
CONSIDERATIONS FOR CHOOSING MFSQL CONNECTOR AS A M-FILES INTEGRATION PLATFORM

Published by Laminin Solutions : 1 Oct 2017

Table of Contents

Synopsys 3
Background 3
MFSQL Connector Benefit 3
Why choose MFSQL Connector 4
Key considerations 4
Related applications data store 4
Method of data exchange 5
Timing of action 5
Available skills 6

CONSIDERATIONS FOR CHOOSING MFSQL CONNECTOR

Scope and complexity 6
Refresh external value lists and object types only 7
Extendibility 8
Code operation 8
Security considerations 9
APPENDIX A - M-FILES DEVELOPMENT (extract from http://developer.m-files.com/) 10
Built in functionality: 10
APIs 10
COM/.NET API 10
The M-Files Web Service (MFWS) 11
Frameworks 11
User Interface Extensibility Framework (UIX) 11
Vault Application Framework (VAF) 11
Development Licences 12
APPENDIX B - MFSQL CONNECTOR (http://tinyurl.com/mfsqlconnector) 13
Business cases 13
Certification 13
Components 13

CONSIDERATIONS FOR CHOOSING MFSQL CONNECTOR

Synopsys

M-Files is a powerful platform to create metadata based control of data and content. It provides a framework for interacting with external applications and systems. MFSQL Connector is an approved M-Files Add-in to enable extensive interaction between M-Files and SQL without mastering the M-Files APIs. It specifically geared to users with SQL Server capabilities or requirements to interact with third party systems; to performed advanced and large scale metadata refinement and cleansing and reporting using M-Files metadata.

There are many alternative methods to build integration between M-Files and third-party systems. Each method will have its own merits, preferences and implications. This white paper elaborates on considerations which should be considered when selecting MFSQL Connector as a platform for building third party integrations.

The most dominant motivators include: Pre-existing SQL skills and platform in the organization; Benefits for deploying the extensive data management powers of SQL; limited access to DotNet development skills; Using the same data for different applications and use cases.

The whitepaper elaborates on considerations when deciding on a integration strategy.

Background

M-Files provides a broad and powerful set of develop-orientated APIs, development tools, frameworks and methods to extend the standard configuration of M-Files and provide an extended reach into other systems and operations. In addition, M-Files Partners and Certified Application Developers complement and add to this toolset with approved M-Files Add-ins to service specific uses and market segments.

This whitepaper elaborates on the considerations for using the MFSQL Connector framework and tools for integration projects.

For the purposes of this whitepaper, integration is defined as exchanging metadata and or files between M-Files and third-party applications. It considers using M-Files core development technologies and/or MFSQL Connector as the first choice.

Refer to appendix A and B for more information on the M-Files Development Framework (MFDF) and the MFSQL Connector Framework (MFSQL).

MFSQL Connector Benefit

The overarching benefit of using MFSQL Connector is to perform data integration tasks with M-Files without the need to use MFDF and program in .NET. The appropriate MFDF functions and methods is wrapped inside MFSQL to allow a SQL knowledgeable individual to interact with M-Files using T-SQL

procedures. Note that all the M-Files integration functionality included in MFSQL Connector can be accomplished by redeveloping it using the M-Files Development Framework.

Why choose MFSQL Connector

a) MFSQL Connector should be a development option if your company have \ or intend to have SQL skills and use Microsoft SQL databases as part of your data infrastructure.
b) M-Files built in functionality does not cover all types of integration use cases. Some use cases will require either using MFDF and/or M-Files Partner sponsored methods to accomplish the task.
c) Using MFDF for integration requires advanced developer skills with Microsoft .NET, Web Technologies and programming experience. These skills may not be readily available.
d) Using MFDF takes time and effort.
e) Requirements must be clear and changing requirements may also require significant effort. MFSQL provides prototyping and dynamic reconfiguration capabilities for more flexibility and rapid changes.
f) It takes time and effort to master MFDF especially for someone that is not working with .NET on a day to day basis. This may not be cost effective for one off projects.
g) SQL skills within the organisation are often more readily available.

Key considerations

The choice of development approach for integration involves many, often competing, considerations. The considerations below highlight different aspects to consider. The considerations and weight of each consideration, will be unique for every development project.

Note that in all cases MFDF can be used. It is not explicitly stated as an option to be chosen below, unless it is the only option.

Related applications data store

The related development environment is determined by the access and availability of the data store of the existing applications, as well as the new applications that will be developed. For example, the CRM system is in the cloud and the only access to the data is through an API call; the ERP system is on premise and use MS SQL; the intranet website is still to be developed and the database architecture is not yet decided.

Consideration	Choice	Comments
Third-party application uses SQL as its own database	MFSQL Connector	Interacting between SQL Databases is commonplace

CONSIDERATIONS FOR CHOOSING MFSQL CONNECTOR

Consideration	Choice	Comments
Third-party application has a non-accessible database	M-Files Web-Services	Likely to use third-party APIs. Introducing a database may add complexity if it would have no other benefits to the organisation
Mixed environment	MFSQL Connector and Third Party API's into MFSQL Connector DB	The custom .net development will focus on the Third-Party API's rather than mastering both M-Files and Third-Party API's

Method of data exchange

Integration is about getting data (metadata and files) from and to M-Files. Third-party applications can often use a variety of different methods, while some are more restricted.

Consideration	Choice	Comments
Database to Database	MFSQL Connector	SQL is built with data manipulation in mind.
ODBC to Database	MFSQL Connector or M-Files Built in functionality	Built in functionality is limited when dependent objects are involved or data preparation is required
Data saved as a file (CSV, XML, Excel, JSon)	MFSQL Connector or M-Files Special tools	Using SSIS with MFSQL Connector is a powerful way to automate the use of file based imports and exports.
Files on the M-Files Server network	M-Files built in functionality or M-Files special tools	MFSQL Connector use standard M-Files tools for file operations

Timing of action

Considering when a particular action, or its associated actions will take place could set the choice for selecting different types of approaches. For instance, the built in functionality for data exchange is little control over the timing of the integration action.

Consideration	Choice	Comments
Timing of the integration action is not material impact	Built in functionality	This is often the case when importing/exporting data on a time scheduled basis

One integration action is dependent on another	MFSQL Connector	This is particularly important when the metadata of one object is dependent on another to be created in the same session
Integration action is triggered by another operation in the background	MFSQL Connector	This trigger could be third party application related, or based a state change or other change of data
Integration action must be triggered by an event in M-Files	MFSQL Connector	This is often related to a change of state of the object
Integration action must be started by the user in M-Files on demand	MFSQL Connector	An example is that the user want to update information in or from the third party system on demand

Available skills

Any development and long term maintenance of the developed applications require a combination of skills. These skills can be sourced internally or externally. The availability of the required skills, especially internal to the organisation have a major impact on the cost and long term sustainability of the solution.

Consideration	Choice	Comments
Company use SQL internally and have access to SQL resources	MFSQL Connector	Any intermediate SQL skilled resource can use MFSQL Connector. .Net skills are not required.
Company has no or limited .Net resources	MFSQL Connector	Upskilling in SQL for internal maintenance and extending the integration is much less onerous that upskilling in .net.
Company has sufficient skilled .Net resources and no SQL resources	M-Files development tools	This is particularly relevant if the company has no intention to extend the use of SQL as a data management platform.

Scope and complexity

Scope and complexity of the data integration requirement is a major consideration. In many cases the use case can be achieved by using standard built in functionality, or a simple boiler plate based API connection. In other cases, the individual use cases may appear to be simple but if it combined into the current and future functional requirements, it becomes more involved and a different solution may be more appropriate.

CONSIDERATIONS FOR CHOOSING MFSQL CONNECTOR

Consideration	Choice	Comments
The data exchange involves a few simple, straight forward data sets	Use one of the built-in connection methods (ODBC connector; external file import)	Straightforward means no complex object dependencies at create stage; all data is import ready without the need for validation or cleansing
The data exchange takes place dynamically without any source or intermediate databases	M-Files API's	Data can be access directly using the API's.
The data exchange use a source database and requires extended processes before or at import stage	MFSQL Connector	Use SQL for the heavy lifting of data preparation and managing multiple objects
The use case extends to using the metadata for advanced reports	MFSQL Connector	The data is already available in SQL and therefore accessible for any standard report writer utility
The use case extends to metadata cleansing or alignment	MFSQL Connector	SQL provides powerful tools to align and manipulate data. This can be applied to the M-Files data and updated into M-Files on completion of the alignment.

Refresh external value lists and object types only

The built-in functionality provides the ability to refresh external value lists and object types from SQL. The features of this functionality have improved further from version 15.3. Considerations for relying solely on the built-it functionality versus applying MFSQL Connector include the following

Consideration	Choice	Comments
The integration operations include ONLY the exchange of data for some object types of value lists	Use built-in functionality	MFSQL Connector is especially useful to expand the scope of the operations beyond simply exchanging data. The integration project often starts with using the built-in functionality and then migrate into the use of MFSQL Connector when the project becomes more involved and complex.

Dependent objects goes beyond a simple parent child relationship based on ownership relationship	MFSQL Connector	The Built-in functionality will only respond to a parent child relationship if there is a set ownership relationship
The data exchange operation need to trigger other related operations	MFSQL Connector	Conditions, matching, data alignment, secondary operations and many more
The dataset need to be updated in batch	MFSQL Connector	The built-in functionality will update the data when and not provide ability to first prepare a batch of information and then the run the update (e.g. delete only some records; prepare data and when condition is met, then update)

Extendibility

The implications of adding new features to existing integration projects, making changes to M-Files metadata structure or adding new applications to integrate with M-Files must be considered.

Consideration	Choice	Comments
Use case is unlikely to change	Either M-Files API's or MFSQL Connector	Depending on the available skill set
There is no requirement for additional features or applications	Either M-Files API's or MFSQL Connector	Depending on the available skill set
There is a high likelihood of changes to the vault structure or for more applications	MFSQL Connector	Using M-Files API's will necessitate going back to the core development and recompiling the solution. Adding features and additional applications using the MFSQL Connector backbone does not require recompiling core code.

Code operation

This consideration highlights the requirement of the code to run client side or server side.

Consideration	Choice	Comments

User action that must run client side with the result instantly available to the user	M-Files API's or scripting	MFSQL Connector is primarily a server side operation.
User action to start a process that can run in the background (on premise)	MFSQL Connector using the Context Menu module	The context menu provides capability to trigger a SQL procedure to process extended operations in the background
User action to start a process that can run in the background (Cloud)	M-Files API's MFSQL Connector if VPN to cloud is used	MFSQL Connector Context Menu is not available for cloud installations without a VPN connection
The operation runs server side	MFSQL Connector or M-Files API's	MFSQL Connector uses the server side M-Files API's at its core and perform all operations server side.

Security considerations

This section considers the specific requirements around security.

Consideration	Choice	Comments
IT Security policy disallow the use of CLR in SQL	M-Files APIs	MFSQL Connector is dependent on the use of CLR technology. Note that M-Files vault database in SQL also require enabling of CLR.
Data access in SQL must follow the same rules as the security settings in M-Files	M-Files APIs	MFSQL Connector data is protected by access permissions in SQL and the third-party application. It does not inherit the M-Files security profile

APPENDIX A - M-FILES DEVELOPMENT (extract from http://developer.m-files.com/)

M-Files provide a range of frameworks and functionality that have a direct bearing on data exchange use cases.

Built in functionality:

M-Files provides significant built-in functionality which can be used by developers and non-developers to create integrated solutions. M-Files' scripting environment allows VBScript to be executed in response to object or server events, or as objects move through workflows. Objects can be retrieved from remote ODBC-compatible data sources, or Custom External Object Type Data Sources can be created to extend this functionality to other sources, such as web services. Files can also be imported from External File Sources and can import content from XML files produced by various scanning and imaging software.

APIs

M-Files provides two Application Programming Interfaces for developers: the COM/.NET API and the M-Files Web Service (MFWS). The choice of which to use in each scenario will depend upon the technology you are using and the operations that you wish to undertake.

COM/.NET API

- Our most comprehensive API, providing interfaces for both "user" and "administrative" functions.
- Can only be run on Windows, where the M-Files COM object can be made available.
- Supports the same connection protocols as the desktop client.
- Supports the same authentication schemes as the desktop client.
- Can be run in "client" or "server" mode:
    - Client mode requires a vault connection is already set up within the M-Files Desktop Settings, and can show M-Files dialogs such as the metadata card for object creation.
    - Server mode does not require a vault connection to be set up on the host machine, but cannot show M-Files dialogs.
- Requires the same version of the API on the client machine as the server.

> More information is available in the COM API section.

CONSIDERATIONS FOR CHOOSING MFSQL CONNECTOR

The M-Files Web Service (MFWS)

- A REST-like web service that is available from within M-Files Web Access.
- Can be called from any environment that can make HTTP requests (e.g. mobile), and is not limited to Windows operating systems.
- Connections to the MFWS are done via HTTPS.
- Supports most “user” operations, but cannot be used to undertake “administrative” functions.
- Not tied directly to the M-Files Server version.

> More information is available in the REST API section.

Frameworks

M-Files provides two separate frameworks for building applications that run within the M-Files software: The User Interface Extensibility Framework (UIX), and the Vault Application Framework (VAF). The User Interface Extensibility Framework (UIX) is used to create client-side applications that interact with, replace, or react to, the M-Files Desktop client or M-Files Web Access. The Vault Application Framework (VAF) is designed as a replacement for using VBScript within M-Files vaults, allowing the use of .NET code instead.

User Interface Extensibility Framework (UIX)

- Used to create client-side applications.
- Can tailor the user interface, such as changing logos or showing or hiding UI elements.
- Can create buttons and menu items which can react to selected items.
- Can create “dashboards” which are shown on demand, and can be provided with content from M-Files.

> More information is available in the User Interface Extensibility Framework section.

Vault Application Framework (VAF)

- Used to create server-side applications.

- Can be used to execute .NET code in response to object (e.g before an object is checked in) or vault events (e.g. before a view is deleted).
- Can be used to execute .NET code as an object moves through a workflow.
- Can be used to create background operations which execute periodically.
- Can be used to execute .NET code to calculate property values and/or provide property value validation.

> More information is available in the Vault Application Framework section.

Development Licences

Development licences are available to M-Files resellers and members of the Certified Application Provider program. For more information contact your Channel Account Manager or the developer evangelism team. Alternatively, a 30-day trial of M-Files can be downloaded from https://www.m-files.com/en/download-latest-version.

APPENDIX B - MFSQL CONNECTOR (http://tinyurl.com/mfsqlconnector)

MFSQL Connector (The Connector) is a developer framework designed to allow M-Files to interact with Microsoft SQL without having to first apply the M-Files development technologies to create the integration code. The Connector allows for using SQL procedures to perform the integration operations. In the background, the Connector use the standard M-Files API's, Vault Application Framework, and UIX framework as SQL CLR assemblies and M-Files Applications.

Business cases

The Connector targets a range of business cases including but not limited to:

- Integration of M-Files with other applications with two-way integration between M-Files and MS SQL
- Creating M-Files Add-on applications using SQL as the extended database for these applications
- Advanced M-Files metadata cleansing and metadata management
- M-Files metadata reports using any report writer or integrating the reports with in-house web applications

In order to expose M-Files metadata, MFSQL Connector imports data from M-Files using LSConnectWrapper as CLR (Common language Runtime) assemblies in SQL, wrapping the interop.MFilesAPI into SQL. Bi-directionally updates using a range of T-SQL procedures interacts with the assembly to perform the integration processes.

MFSQL Connector is installed on the SQL server into a separate database and does not interact directly with the M-Files Vault database and requires the M-Files client to be installed on the SQL Server.

Certification

MFSQL Connector is a certified M-Files Add-in in terms of the M-Files Certified Application Developer (CAP) program.

Components

Assemblies: These are proprietary programs (.dll) to enable the interaction between M-Files and SQL. The main purpose of the assemblies is to wrap the standard M-Files API's into code classes that allows data to be extracted from M-Files into SQL tables and to update M-Files from the SQL Tables. There are four assemblies included.

CONSIDERATIONS FOR CHOOSING MFSQL CONNECTOR

SQL Connector Tables: These tables contain the metadata of the vault, settings, and transaction logging data. MFSQL Connector tables are all prefixed with “MF” to distinguish them from non-Connector tables in the database.

Metadata tables: The metadata about metadata table contain the information about the structure of M-Files (e.g. classes, properties, valuelists, etc.) Connector Table related to this type of metadata include Object Types; Classes; Properties; Class_Property; Valuelists; Valuelist items; Workflows; Workflow States; User Accounts; Login Accounts

Utility tables: These are supporting tables and include: Datatypes; Settings; Process; DeploymentDetail

Logging Tables: These tables reference processing logs and include: MFLog; UpdateHistory; BatchProcessLog

M-Files Class Tables: These table represent the data in the vault such as Customers, Contacts and any other class in the vault. Class tables are not created by default, but created when the data that is required for the specific application has been determined. The Class Tables therefore represent the metadata of the specific classes of objects that the application will interact with.

SQL Procedure & functions: These T-SQL procedures and functions perform operations on the tables and the assemblies. These procedures are all prefixed with “spMF”.

Context Menu: Execute SQL Procedure or URL from within the M-Files desktop context and include:

- In M-Files
  - M-Files Vault Application
  - UIX components
  - Configuration Content Package
- In SQL
  - Context Menu Tables
  - Context menu Procedures

- More information is available in the Connector Guide
- https://Lamininsolutions.com